



Universidade de São Paulo

Instituto de Ciências Matemáticas e de Computação

Engenharia de Computação - 2023.2

SSC0640 - Bases de Dados

Exploração de planetas

Subtema: Sistema para planejamento de missões espaciais

{00}

Docente: Profª Drª Elaine Parros Machado de Souza

Estagiário PAE: André Moreira Souza

Beatriz Lomes da Silva	12548038
Hugo Hiroyuki Nakamura	12732037
Isaac Santos Soares	12751713
João Pedro Gonçalves Ferreira	12731314
Nicholas Estevão P. de O. Rodrigues Bragança	12689616

1. Descrição do Problema e dos Requisitos de Dados:

A aplicação desenvolvida, idealizada em uma realidade fictícia, visa armazenar dados relacionados à exploração planetária, com o objetivo de facilitar o planejamento de missões espaciais. O usuário da plataforma é um gerente de operações humano, capaz de consultar dados do banco para planejar uma missão espacial, verificando sua viabilidade técnica.

Assim, o usuário do sistema deve ser capaz de manipular dados de planetas, sistemas planetários, civilizações, naves espaciais, bases espaciais, rotas interplanetárias e ameaças, através de operações de cadastro, atualização e remoção. Todos esses dados são necessários para planejar diferentes tipos de missões espaciais, assim como consultar o histórico de missões já realizadas e os dados que ela gerou ou utilizou. Para o modelo, assume-se que todas as naves são capazes de partir de um ponto a outro na *velocidade da luz*.

Em seguida, estão descritos todos os conceitos usados para a aplicação.

Um **sistema planetário** é um conjunto maior e mais complexo de configuração de vários *planetas*. Ele é identificado através de seu nome único, juntamente com a galáxia em que está inserido. Além disso, ele tem uma idade estimada, medida em anos, e possui uma – ou mais – estrela central de um tipo específico, como anã branca, anã vermelha, estrela de Quarks etc. Para chegar a um sistema planetário, é preciso percorrer uma certa distância em relação ao Sistema Solar, medida em anos-luz.

Um **planeta**, por sua vez, é um possível alvo de missões espaciais, possuindo condições atmosféricas e geológicas (composição da atmosfera, temperatura global média, pressão atmosférica e clima) e satélites naturais. Ele pode apresentar várias bases, criadas pelos humanos em inúmeras missões de colonização anteriores. Eles são identificados, univocamente, por um único *sistema planetário*, em que estão localizados, juntamente a um nome. Existem também **recursos naturais**, que devem ser possuídos por algum planeta mas podem estar presentes em vários outros planetas e ser explorados por várias missões, identificados unicamente por um certo tipo e um certo nome. No sistema, também é

registrada a estimativa de quantidade de recursos naturais presentes no planeta. É importante destacar que um planeta possui várias rotas para vários outros planetas.

Base espacial é um centro de operação presente em alguns planetas. Além de ser o destino de várias missões, ela também é responsável por criá-las e defini-las. A base tem um nome, que a identifica univocamente, junto ao único *planeta*, onde ela está localizada. Para comunicar-se com a base, é preciso estabelecer o canal de comunicação, em uma certa frequência. Ele nos garante a conexão com a base desejada, para que seja possível a troca de informações entre as colônias, tanto no processo de sua formação, quanto após esse processo. Além disso, ela pode ser definida pelo seu tipo, podendo ser aquática, terrestre ou aérea, e também pela sua capacidade. As bases espaciais possuem, internamente, fábricas de naves, capazes de construir novas bases e repará-las ao fim de cada missão.

Uma **missão espacial** é uma viagem para um planeta destino, a fim de atingir um objetivo específico. No sistema, é preciso informar uma descrição do objetivo almejado, assim como o tamanho da tripulação, a duração estimada e a data de fim. Além disso, todas as missões são identificadas unicamente por um nome, data de início e pela *base espacial* de origem. Elas são criadas considerando o nível de periculosidade apresentado pelo trajeto e pelo planeta destino, sendo definidas entre seguro, baixo, moderado, significativo, grave e crítico. As missões possuem várias **rotas** definidas, ordenadamente, entre dois *planetas* distintos que a identificam, separados por uma distância. Consequentemente, uma mesma rota pode ser utilizada por várias missões espaciais. A missão é criada considerando seu objetivo, isto é, se é voltada para exploração, comércio ou colonização. Todas as missões, obrigatoriamente, originam-se em uma única base e destinam-se, também, a uma única base, mas de outro planeta, exceto a missão de colonização, a única que não parte para uma base espacial.

Uma **missão de exploração** tem como objetivo obter uma certa quantidade de recursos naturais por meio da exploração de um planeta já colonizado. Para isso, deve-se levar equipamentos específicos para realizar a extração dos recursos almejados. A constante extração pode impactar negativamente o planeta e, portanto, deve-se analisar o nível de impacto ambiental que a missão oferece.

A **missão de colonização** busca criar uma base em um único planeta não colonizado por humanos, ou seja, que não possui uma base espacial. A missão só pode ser realizada caso não haja civilizações hostis no destino, pois a colonização ocorre através de acordos pacíficos e mutuamente benéficos.

A **missão de comércio** é realizada para o comércio de alguns produtos com algumas civilizações amistosas habitantes do planeta. Dessa forma, estabelece-se um tipo de comércio – venda ou compra – de uma certa quantidade de produtos. É importante que cada missão de comércio tenha uma verba máxima definida, que será investida na compra dos produtos desejados e que não poderá ser ultrapassada.

Os **produtos** são itens de origem animal, vegetal ou mineral, que possuem nome, criado pelos humanos, um preço, em *spacies*, a moeda do universo, e um peso, em toneladas. Todo produto possui uma breve descrição fornecida pelo usuário, como a sua relevância, importância, utilidade etc. Os produtos são identificados univocamente pelo seu nome e sua origem. Um certo tipo de produto, que pode pertencer a várias civilizações, pode ser comercializado em várias missões de comércio.

A **nave espacial** é o meio de transporte utilizado na viagem interplanetária, oriunda de uma única base. As naves possuem espaço limitado, o que permite um número máximo de tripulantes. Cada viagem desgasta a nave em um certo nível, e, obviamente, uma nave não deve executar uma missão que não é capaz de suportar. Por esse motivo, as naves apresentam uma resistência, que mede a durabilidade máxima da nave. Além disso, cada nave possui uma localização atual em uma base, a menos que ela esteja sendo usada em alguma missão. Devido aos diferentes objetivos de missões, naves específicas precisam ser utilizadas, e, dessa forma, são separadas pela classe, um identificador junto ao seu nome e a *base espacial* em que foi construída. Uma missão de objetivo comércio precisa de uma nave da classe **cargueiro**, que possui uma capacidade máxima, em toneladas, de produtos. Já uma missão de objetivo colonização necessita da nave **colonizadora**, única capaz de visitar planetas sem uma base espacial. Por fim, uma missão de objetivo de exploração precisa de uma nave da classe **científica**, que também

possui uma capacidade máxima para a coleta de itens. Todas as naves podem executar várias missões, mas várias naves não podem executar a mesma missão.

Dentro das condições estabelecidas em cada planeta, é possível haver a presença de **civilizações**. Elas podem habitar vários planetas, mas originar-se em apenas um deles. Dessa forma, um planeta pode abrigar e ser o berço de várias civilizações. Além disso, elas têm uma categoria, de acordo com a escala de Kardashev, como do tipo I, que utilizam apenas a energia proveniente do próprio planeta; tipo II, que também utilizam a energia da estrela em que orbitam; e tipo III, que utilizam a energia proveniente de toda a galáxia. Considera-se que toda civilização tem um nome que é único no sistema, juntamente à *categoria* e ao *planeta* de origem. Ainda, podem ser classificadas como uma civilização amistosa, quando tem um bom relacionamento com humanos, ou hostil, caso contrário. **Civilizações amistosas** são aquelas que não oferecem ameaças, possibilitando tratativas comerciais, através de algumas missões. Por outro lado, as **civilizações hostis** oferecem ameaças e podem impedir a realização das missões. No geral, toda civilização pode ser uma possível uma ameaça, entretanto, a ameaça só é relevante quando a civilização é hostil. Além disso, independente da sua compatibilidade com os humanos, elas possuem vários produtos para se comercializar.

Por fim, as **ameaças** oferecem um nível de perigo à missão, descritas por uma descrição e identificadas por um único nome. Dentro do sistema, as ameaças estão presentes em vários planetas e em várias rotas, assim como um planeta e uma rota possuem várias ameaças. As que são referentes aos planetas podem variar desde uma atmosfera rarefeita até a forte atividade sísmica. Já as referentes às rotas envolvem possíveis colisões com corpos celestes ou a passagem da nave por um caminho instável. Além disso, como já descrito anteriormente, várias civilizações podem ser uma ameaça, caso sejam hostis, e comprometer a integridade física da nave e dos tripulantes.

2. Restrições de integridade:

Deve ser utilizado o tipo (classe) de nave correto para cada tipo de missão. Se o objetivo da missão é criar uma nova base (Colonização) então a nave utilizada deve ser uma Nave Colonizadora. Analogamente, missões de Exploração devem ser realizadas por Naves Científicas e missões de Comércio devem ser realizadas por Naves de Carga.

A missão deve iniciar e terminar em uma base a menos que o objetivo da missão seja criar a base. Em outras palavras, missões de Exploração e Comércio devem obrigatoriamente ter uma base de destino (não podem finalizar em um planeta inabitado). Missões de Colonização não tem uma base de destino mas, ao final da rota, constroem uma nova base no planeta que chegaram.

Para a existência de uma missão espacial deve ter um objetivo e uma nave espacial, podendo utilizar apenas uma nave e ter apenas um objetivo.

Deve ser permitido que uma civilização não habite mais o planeta em que se originou, porém, o planeta de origem ainda deve ser armazenado. A civilização pode habitar diversos planetas, no entanto só pode ter sido originada por um planeta.

O comércio só deve ser permitido com civilizações amistosas. Em missões de comércio, diversos produtos podem ser comercializados com diversas civilizações.

A classificação da civilização em hostil ou amistosa pode variar, por exemplo, se a ameaça que ela representa for excluída, assim uma civilização que era hostil se torna amistosa; o contrário também pode ocorrer por meio do cadastro de ameaças.

A nave espacial pode estar associada a diversas missões de várias bases espaciais, quando disponível, e só pode ter sido construída por uma base espacial.

O planeta pode ter várias bases espaciais, porém a base espacial está em apenas um planeta. A base só pode ter sido construída em uma base espacial. Só é permitida a criação de uma base espacial em um planeta já associado a um sistema planetário.

Ameaças podem ser associadas a rotas, planetas e a civilizações. Uma ameaça é associada a uma rota quando ela ocorre no caminho entre os dois planetas da rota. Uma ameaça é associada a um planeta quando ela está no planeta e afeta todas as missões e rotas que passem por ele. Uma ameaça é associada a uma civilização quando a civilização é hostil e, portanto, está presente em todos os planetas que essa civilização habita.

3. Possíveis Inconsistências:

A nave utilizada em uma missão deve ter resistência suficiente para suportar todas as ameaças existentes nas rotas e nos planetas que a missão envolve. Se a rota da missão tem um alto grau de periculosidade ela **exige** uma nave que suporte as ameaças. A soma dos níveis das ameaças em uma missão não deve ultrapassar a resistência máxima da nave. Caso contrário, será necessário parar em uma base antes para reparar a nave (realizar duas missões separadas). As civilizações hostis também apresentam ameaças à missão; logo, se a missão passa por um planeta habitado por uma civilização hostil a nave deve conseguir suportar a ameaça da civilização também. O MER não consegue garantir essa consistência.

Ciclo (Planeta - tem rota - Planeta): O sistema não deve permitir registrar rotas do planeta nele mesmo, tais caminhos não convém no contexto da aplicação. O MER não consegue garantir essa consistência.

A missão nada mais é do que um conjunto de rotas que juntas formam um caminho do planeta/base de origem ao planeta/base de destino. Deve-se assegurar a coerência de que o caminho traçado coincide com o planeta/base de origem e destino. O MER não consegue garantir essa consistência.

O sistema armazena a localização atual das naves, calculada por meio do histórico de missões. Deve-se garantir que ao ser utilizada em uma missão a nave esteja localizada atualmente na base/planeta de origem. O MER não consegue garantir essa consistência.

Também relacionado com a localização das naves, o MER permite que uma nave que já está sendo utilizada em uma missão seja cadastrada em outra que ocorre simultaneamente. Assim, é necessário que a aplicação faça a verificação de disponibilidade da nave, ou seja, se ela não está sendo usada em nenhuma outra missão no intervalo de datas desejado.

Missões de Comércio possuem um orçamento máximo, deve-se garantir que o gasto total da missão não extrapole esse valor. Por gasto total entenda-se a soma dos gastos (compras) menos a soma dos lucros (vendas). O MER não consegue garantir essa consistência.

Missões de comércio só podem ser realizadas com civilizações amistosas, aquelas que não apresentam ameaças. O MER não consegue garantir essa consistência, pois a especialização da civilização é feita com base em um atributo derivado, logo pode variar; se o relacionamento de comércio fosse vinculado apenas com a civilização não hostil poderiam ocorrer problemas se a classificação da civilização trocar.

Ciclo (Civilização - possui - Produto - comercializado por - Civilização): É necessário também que o sistema só permita que a civilização venda os produtos que ela possui. O MER permite que uma civilização comercialize produtos que ela não possui, isso deve ser impedido em aplicação.

Ciclo (Civilização - comercializa na - Missão de Comércio - que possui - Rota - Planeta - habitado por - Civilização): O MER permite que uma missão de comércio comercialize produtos com uma civilização que não habita nenhum dos planetas da rota da missão, isso deve ser impedido em aplicação.

Ciclo (Civilização - Ameaça - Planeta - habitado por - Civilização): O MER permite que uma civilização ameace um planeta que ela não habita. Também permite que a civilização ameace uma rota entre planetas que ela não habita.

O MER permite o comércio com civilizações hostis, o que não está previsto na aplicação.

Ciclo (Planeta - tem - Base Espacial - que é origem da - Missão Espacial - Rotas - passam por - Planeta): O MER permite que uma missão espacial tenha uma base de origem que está localizada em um planeta que não está na rota da missão. Deve ser garantido em aplicação que a base de origem está sempre na primeira rota da missão.

Ciclo (Planeta - tem - Recurso Natural - explorado por - Missão Espacial - Rota - passa por - Planeta): O MER permite que uma missão de exploração explore um recurso natural que não existe em nenhum dos planetas da rota da missão.

Ciclo (Missão de Colonização - coloniza - Planeta - que tem - Rota - usada pela - Missão de Colonização): O MER permite que uma missão de colonização colonize um planeta que não está na rota da missão. A aplicação deve garantir que o planeta colonizado seja o último planeta da rota da missão (destino).

Ciclo (Base Espacial - é destino - Missão Exploração/Comércio - que tem - Rotas - passa por - Planeta - que tem - Base Espacial): O MER permite que as missões de exploração e comércio tenham como destino uma base espacial que não está localizada em nenhum dos planetas da rota da missão. A aplicação deve garantir que a base de destino esteja no último planeta da rota da missão (destino).

4. Interesse da aplicação:

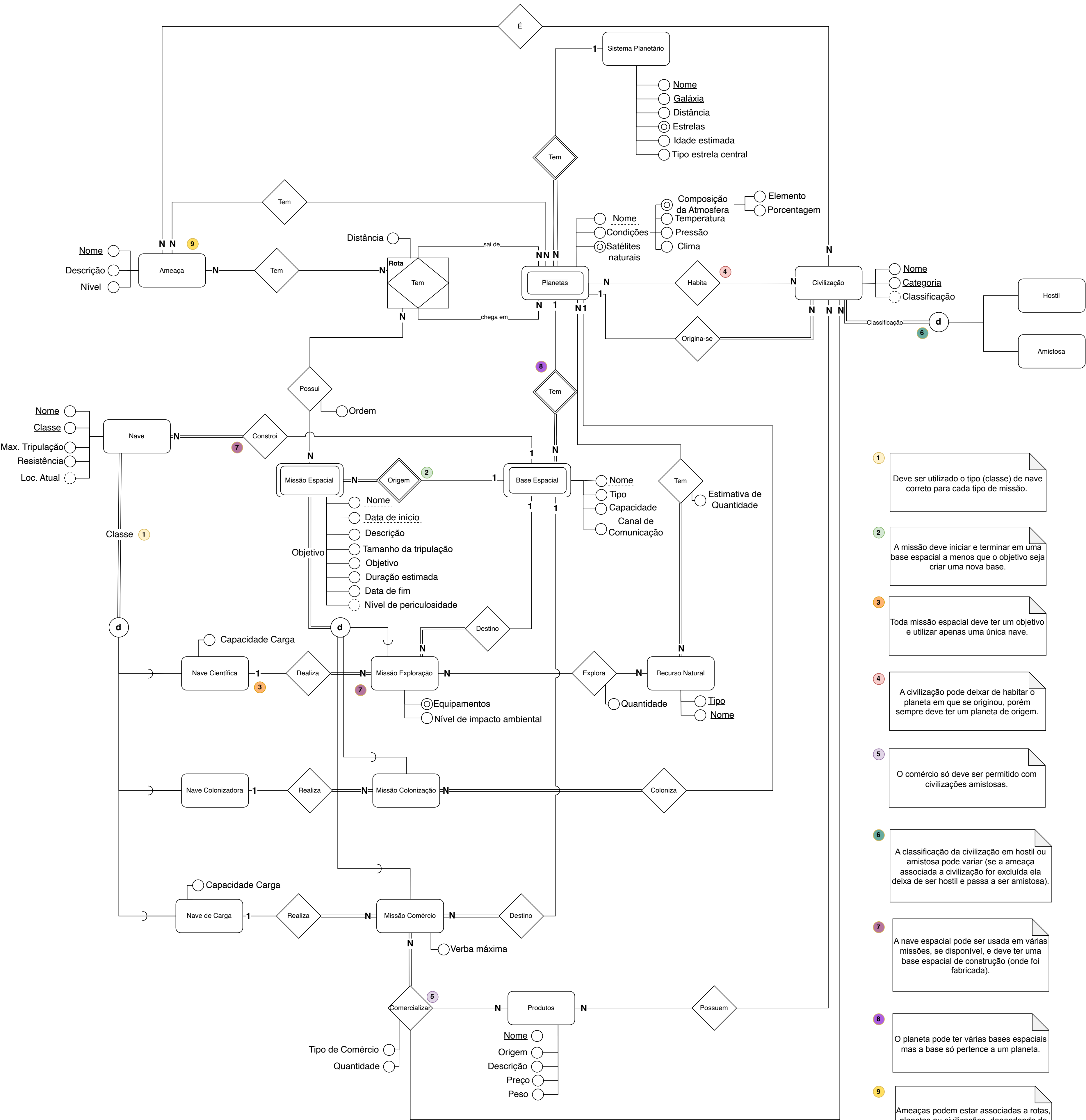
Gerente de operações que planeja, confere missões que já foram feitas e verifica a viabilidade de possíveis novas missões espaciais. Registra dados coletados em expedições já realizadas.

5. Principais Funcionalidades:

- Cadastrar, atualizar, remover planetas, sistemas planetários, civilizações, missões espaciais, naves espaciais, bases espaciais, rotas entre planetas e ameaças.

- Verificar se uma dada civilização é compatível com a humana (não apresentar ameaças).
- Verificar que tipos de base cada planeta comporta (determinado com base nas condições ambientais)
- Lista rotas de expedições mais seguras.
- Lista rotas de expedições mais perigosas.
- Criar missões espaciais (que percorrem determinada rota que iniciam e terminam em uma base).
- Verificar as ameaças numa possível missão.
- Verificar se a nave é capaz de percorrer a rota (se a nave suporta o nível de periculosidade da viagem).
- Traçar uma rota com menor índice de ameaças.

6. Projeto Conceitual: Modelo Entidade-Relacionamento (MER-X)



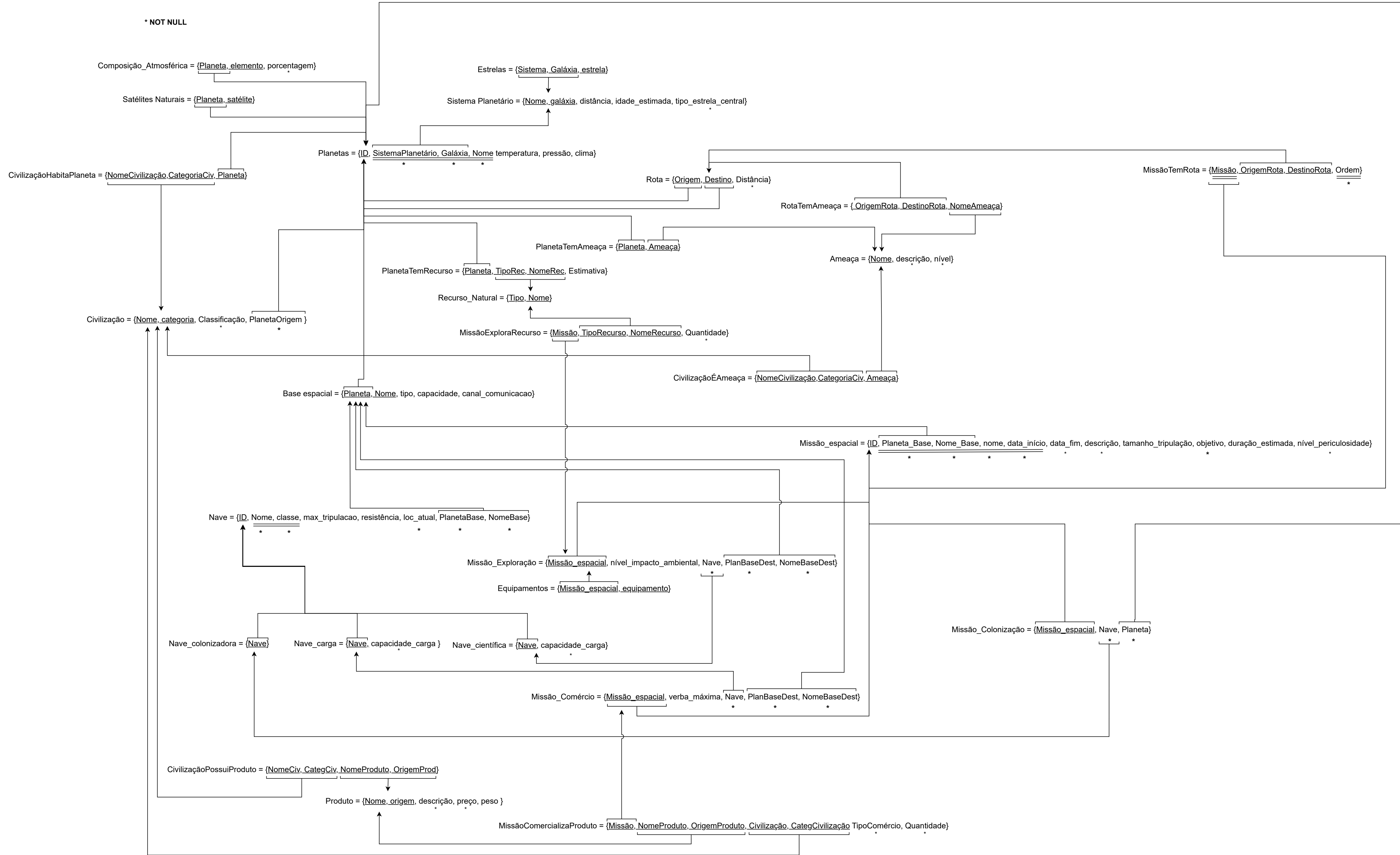
7. Correções Parte 1:

Na seção de Possíveis Inconsistências foram inseridas as entidades e os relacionamentos envolvidos nos ciclos descritos.

No Modelo Entidade Relacionamento foi corrigida para atributo multivalorado a notação do atributo Equipamentos da entidade Missão Exploração.

Sobre a nave espacial ser ou não uma entidade fraca da Base: foi uma escolha de projeto pela semântica do relacionamento. No mapeamento para o Modelo Relacional as duas soluções seriam equivalentes. Porém optou-se por colocar o relacionamento "constrói" com participação total assim como decidiu-se mapear que uma Civilização "origina-se" em um planeta como um relacionamento com participação total e não uma entidade fraca. Foi uma escolha de modelagem baseada na semântica do sistema.

8. Modelo Relacional



9. Justificativas Modelo Relacional:

Nesta seção serão apresentadas as justificativas para as escolhas de mapeamento do Modelo Relacional. Note que algumas das soluções alternativas (item 9.1) foram agrupadas por tipo de relacionamento e apresentadas no final da seção para evitar repetição no texto.

- **Atributos multivalorados: Estrelas (Sistema Planetário), Satélites Naturais (Planeta), Composição Atmosférica (Planeta) e Equipamentos (Missão Exploração):**

Nesses quatro casos há grande variabilidade na quantidade de valores armazenados, por isso todos foram mapeados em tabelas separadas.

O atributo Estrelas foi mapeado na tabela Estrelas para evitar vários valores nulos. Se fosse mapeado como colunas da tabela Sistema Planetário a grande maioria teria apenas uma estrela e os outros valores seriam nulos.

Analogamente, há planetas com 1 satélite natural (ex: Terra) e planetas com 92 satélites naturais (ex: Júpiter).

Com relação à composição atmosférica não há como prever uma dada quantidade de componentes químicos que são relevantes de serem mencionados em todos os planetas. Não convém registrar, por exemplo, os 3 componentes químicos mais presentes nas atmosferas já que pode, por exemplo, existir um componente altamente tóxico que convém ser mencionado, porém sua participação percentual na atmosfera não é tão alta.

A justificativa para o mapeamento dos equipamentos é a mesma que a da composição atmosférica, não há como prever quantos equipamentos uma missão irá precisar cadastrar no sistema.

Solução adotada: Mapear os atributos multivalorados em tabelas separadas.

Vantagem: Evita valores nulos.

Desvantagem: Resulta em novas tabelas que implicam na necessidade de junção para consultar o valor dos multivalorados.

Solução alternativa: Mapear os multivalorados na tabela de origem colocando uma quantidade limitada de campos.

- **ID sintético Planeta, Nave e Missão Espacial:**

Foi criado um ID sintético para Planeta pois é uma entidade fraca identificada por três strings (Sistema Planetário, Galáxia e Nome Planeta) que tem muitos relacionamentos com outras entidades. O uso do ID sintético resultará em grande economia de

armazenamento pois evitará a replicação dessas strings nas tabelas dos relacionamentos com essa entidade.

Analogamente, as Naves são identificadas pelo seu nome e classe (duas strings). Criamos um ID sintético inteiro para evitar a replicação dessas strings nas tabelas da especialização e nos relacionamentos, economizando armazenamento.

As Missões Espaciais são identificadas pelo ID do planeta da base de origem, pelo nome da base de origem, pelo nome da missão e pela data de início desta. Criamos um ID sintético inteiro para evitar a replicação desses dados nas tabelas da especialização e nos relacionamentos, economizando armazenamento.

Solução adotada: Criação de ID sintético inteiro

Vantagem: Economia de armazenamento pois reduz o tamanho das chaves usadas como chaves estrangeiras nos relacionamentos em outras tabelas.

Desvantagem: Afeta o desempenho das consultas pois necessita de mais junções. Por exemplo, para saber o nome do planeta em que está uma base espacial agora é necessário fazer uma junção com a tabela de planetas (antes essa informação estava na própria chave)

Solução alternativa: Uso das chave primária original composta.

- **Atributos derivados: Classificação (Civilização), LocAtual (Nave) e Nível Periculosidade (Missão Espacial):**

Nesses três casos os atributos derivados foram mapeados no relacional e cadastrados no banco. Eles são úteis na aplicação a só precisam ser atualizados pontualmente e não para o conjunto de entidades completo. Armazená-los no banco é mais conveniente do que calcular sobre demanda.

A classificação da Civilização é útil para facilitar a verificação se pode ou não realizar comércio com uma civilização (é amistosa ou não) e só precisa ser atualizado quando altera-se as ameaças associadas a uma dada civilização.

A LocAtual de uma Nave só precisa ser atualizada ao término da missão que ela está realizando.

O nível de periculosidade da missão só precisa ser alterado quando são modificadas as ameaças associadas às Rotas, Civilizações e Planetas envolvidos naquela missão. Uma outra consideração importante é que as ameaças são dinâmicas, isto é, podem existir hoje mas no futuro deixarem de existir. A escolha de mapear o atributo nível de periculosidade permite manter um histórico de nível no caso de missões que já terminaram, mesmo que as ameaças que existiam no período da missão não existam mais. Ademais, isso implica que o

atributo nível de periculosidade só precisa ser atualizado para as missões atuais e futuras; o que reduz significativamente o esforço de manter esse atributo atualizado.

Solução adotada: Mapear os atributos derivados em sua tabela de origem

Vantagem: Facilidade de obter esses atributos nas consultas

Desvantagem: Necessidade de manter atualizado

Solução alternativa: Não mapear a calcular sob demanda.

- **Especializações de Missão Espacial e Nave:**

Em ambos os casos mapeamos tanto as entidades genéricas quanto as específicas. Foi necessário mapear as genéricas por causa dos vários relacionamentos que elas participam; para evitar a replicação desses relacionamentos, já que muitos deles resultam em novas tabelas.

Foi necessário mapear as entidades específicas também por causa dos relacionamentos que elas participam. Sem mapear as entidades específicas não seria possível garantir que cada tipo de missão está usando o tipo de nave correspondente: Nave Colonizadora para Missão de Colonização, Nave de Carga para Missão de Comércio e Nave Científica para Missão de Exploração.

Os atributos de critério (classe e objetivo) foram cadastrados no banco, pois raramente uma entidade específica trocará de categoria.

Devido a essa escolha de mapeamento não foi possível garantir a especialização total prevista no MER. Mesmo com not null nos atributos de critério não há como garantir que o entidade também será cadastrada na tabela do conjunto de entidades específicas. De forma análoga, não garante-se a disjunção pois, apesar do atributo de critério ser monovalorado, não há como impedir que a chave estrangeira seja usada em mais de uma tabela das entidades específicas.

Solução adotada: Mapear as entidades genéricas e as específicas

Vantagem: Implementar os relacionamentos como previsto no MER, garantindo consistência entre tipo da nave e tipo da missão. Evitar a duplicação dos relacionamentos genéricos que resultam em muitas outras tabelas.

Desvantagem: Não garante especialização total nem disjunção.

Solução alternativa: As principais alternativas eram mapear somente as entidades específicas, somente as genéricas, ou mapear tudo em uma única tabela.

- **Especialização de Civilização em Amistosa em Hostil:**

Como as entidades específicas não possuem relacionamentos nem atributos específicos, optou-se por mapear apenas a entidade genérica com atributo de critério (classificação) not null que garante a especialização total prevista.

Essa forma de mapeamento evita a criação de novas tabelas e permite que uma civilização troque de categoria facilmente, como previsto na descrição do sistema.

Como o atributo de critério é também um atributo derivado do relacionamento com a entidade Ameaças é possível que a classificação da Civilização varie bastante. Nesse contexto, a implementação dos conjuntos de entidades específicas separadamente não é viável, pois os dados teriam que ser copiados de uma tabela para outras frequentemente.

Além disso, o atributo de critério determina se é permitido ou não o comércio com uma dada civilização; optou-se por salvar esse atributo no banco dada sua utilidade e atualização pontual.

Solução adotada: Mapear somente a entidade genérica com atributo de critério.

Vantagem: Evita a criação de novas tabelas, permite facilmente a mudança da classificação da civilização, garante disjunção e especialização total.

Desvantagem: Para saber a classificação da civilização é necessário consultar o atributo de critério pois as duas possíveis classificações estão salvas na mesma tabela (dados misturados).

Solução alternativa: As principais alternativas eram mapear somente as entidades específicas, somente a genérica ou mapear a genérica e as específicas.

- **Planeta entidade fraca de Sistema Planetário:**

As duas entidades foram mapeadas separadamente, pois a cardinalidade do relacionamento é 1-N. A alternativa de mapear tudo em uma tabela implicaria em replicar diversas vezes os dados do Sistema Planetário, o que ficaria mais complicado por causa dos multivalorados Estrelas e Satélites Naturais.

No Planeta foi colocada chave estrangeira not null que referencia Sistema Planetário. O not null garante que todo Planeta pertence a um Sistema Planetário.

A chave primária do Planeta também envolve seu nome, dessa forma, garante-se o N da cardinalidade, pois um Sistema Planetário pode dar origem a vários Planetas com nomes diferentes, mas não podem existir dois planetas com o mesmo nome no mesmo Sistema Planetário.

Solução adotada: Mapear as entidades separadamente vinculadas por chave estrangeira

Vantagem: Garante consistência e evita redundância, economizando armazenamento.

Desvantagem: Gera chave primária semântica muito grande que implica na necessidade de um ID sintético para Planeta.

Solução alternativa: Mapear tudo em uma tabela única repetindo os dados dos Sistemas Planetários.

- **Relacionamento N-N de Ameaça com as entidades Rotas, Planeta e Civilização:**

Esses três relacionamentos N-N foram mapeados em tabelas separadas para garantir a cardinalidade. Não há alternativas de mapeamento.

- **Agregação Rota e relacionamento tem rota:**

O relacionamento "tem rota" não foi mapeado. Foi criada apenas a tabela para a agregação Rota, que é identificada pelo par de planetas e que tem determinada distância. Nesse caso não há necessidade de mapear o relacionamento pois não há nenhum atributo do relacionamento nem da agregação.

Solução adotada: Mapear somente a entidade agregada

Vantagem: Economia de armazenamento pois evita uma nova tabela e evita junção.

Desvantagem: Não há.

Solução alternativa: Mapear o relacionamento "tem rota" e a entidade agregada Rota.

- **Relacionamento N-N Civilização habita Planeta:**

Foi mapeado em tabela separada para garantir a cardinalidade. Não há alternativas de mapeamento.

- **Relacionamento 1-N Civilização origina-se em Planeta:**

Foi mapeado por meio de chave estrangeira com not null na tabela Civilização. Como foi mapeado na tabela Civilização a cardinalidade 1-N é garantida e o not null garante a participação total prevista, assim, toda Civilização tem um planeta de origem.

Solução adotada: Mapear chave estrangeira na tabela Civilização referenciando Planeta

Vantagem: Garante Participação Total e não necessita de junção.

Desvantagem: Não há.

Solução alternativa: Criar uma nova tabela (como se fosse N-N).

- **Relacionamento N-N Missão Espacial possui Rota:**

Por ser N-N foi mapeado em uma nova tabela para garantir a cardinalidade; não há alternativas de mapeamento. A chave primária da tabela é o ID da Missão e o ID da Rota. O atributo do relacionamento "ordem" foi mapeado como "not null". Também foi criada uma chave secundária que envolve o ID da Missão e o atributo ordem, essa chave garante que, em uma mesma missão, não será possível cadastrar duas rotas com a mesma ordem. Isso é necessário já que a ordem das rotas define o trajeto que a missão irá percorrer, essa chave ajuda a reduzir inconsistências já que não será possível passar por duas rotas ao mesmo tempo (mesma ordem).

- **Relacionamento 1-N Base Espacial constrói Nave:**

Solução Adotada: Foi adicionado na Nave um atributo de Base espacial, que refere-se a Base em que a nave foi construída. Esse campo recebeu "*", que implica em um campo não nulo, para garantir a participação total da base em naves espaciais.

Vantagens: Dado que toda Nave cadastrada precisa estar associada a uma Base, mapear dessa forma irá garantir que o campo não será nulo ao cadastrar uma inserção, ou seja, não será possível ter naves cadastradas que não estejam associadas a uma Base.

Com a chave estrangeira da Base Espacial associada permite maior facilidade de consulta das naves associadas a determinada base. Essa adição também facilita a compreensão de como a nave está relacionada com a Base.

Desvantagens: Para o mapeamento desse relacionamento não há nenhuma desvantagem evidente.

- **Relacionamento 1-N Planeta tem Base Espacial:**

Solução Adotada: Para existência de uma Base Espacial ela deve estar associada a uma planeta, por ser uma entidade fraca de um. Dado essa limitação foi mapeado de forma que adiciona-se na Base um campo planeta, e essa chave estrangeira planeta compõe a chave primária da Base Espacial.

Vantagens:

Uma das principais vantagens desse método é a facilidade que ele permite em acessar e fazer a exclusão de inserção de uma base associada a um determinado planeta.

Como a existência de uma Base espacial é dependente de um Planeta, ao excluir um é necessário deletar todas Bases conectadas. Pelo campo já estar inserido na própria tabela há uma forma de consulta mais direta e prática.

Pela chave estrangeira ser parte da chave primária facilita compreender a dependência do Planeta na base de modo que o relacionamento seja mais evidente. E também, já que a chave estrangeira faz parte da chave primária já existe uma garantia de que o campo planeta não vai ser nulo.

Solução Alternativa:

Para esse relacionamento não teria outro mapeamento viável a não ser a adição da chave estrangeira Planeta, já que há sua necessidade na composição do identificador da entidade Base Espacial.

Se fosse adotado o método de mesclar em uma tabela só os campos do Planeta e da Base iria gerar vários problemas como complexidade, redundância e desperdício de espaço. Pode existir Planetas que ainda não possuem Base espacial, o que resultaria em um muitos campos nulos, e também para cada nova base de um planeta teria que ser cadastradas informações desse planeta, resultando em uma duplicação dos dados.

- **Relacionamento N-N Planeta tem Recurso Natural:**

Solução Adotada: Criou-se uma nova tabela onde adicionou-se os campos de identificação do planeta, os campos de identificação do Recurso natural, e também o atributo quantidade que é referência ao relacionamento dessas duas entidades. Essa nova tabela criada adotou como chave primária os campos identificadores do recurso natural e o identificador do planeta.

Vantagens:

Garante a adição do atributo apenas no Relacionamento entre entidades. O atributo só faz sentido ser referente ao relacionamento, pois não faz sentido adicionar nas entidades Planeta e nem Recurso Natural, já é referente a disponibilização de um recurso-planeta e não ao total do recurso de modo geral e nem da quantidade de um planeta.

Ao criar uma nova tabela e cadastrar como identificador único o ID do planeta os atributos de identificação do recurso fazendo com que o par Planeta-Recurso não se repita, assegurando não repetição de informações e relacionamentos.

- **Relacionamento 1-N Missão Espacial tem origem em Base Espacial:**

Solução Adotada: Para esse relacionamento foi adicionado os atributos identificadores referente a base espacial na tabela de Missão Espacial. Como a Missão espacial é dependente, ou seja, entidade fraca da Base espacial, as chaves externas adicionadas irão compor a chave primária da tabela.

Vantagens:

A solução adotada garante que toda Missão Espacial cadastrada esteja associada a uma Base espacial, e assim assegurando a participação total da Base Espacial na missão.

Também há a vantagem de maior facilidade de consulta de todas missões associadas a uma determinada Base.

Para o caso de entidades dependentes não há outra solução a não ser adicionar chave estrangeira na entidade fraca, considerando que sua chave primária necessita desse campo para ter um identificador único.

- **Relacionamento 1-N Missão Exploração/Comércio tem destino em Base Espacial:**

Solução Adotada: Para mapear esse relacionamento adicionou-se os atributos identificadores referente a base espacial na tabela de Missão Espacial, e também adicionou-se um “*” abaixo do campo identificador para referenciar a necessidade dele não ser nulo.

Vantagens:

As vantagens desse modo de mapeamento para esse relacionamento são as mesmas que a do Relacionamento 1-N Missão Espacial tem origem em Base Espacial.

- **Relacionamento 1-N Nave Científica/Colonizadora/Carga realiza Missão Exploração/Colonização/Comércio:**

Solução Adotada: Nos três relacionamentos foi adotado o mesmo método de adicionar o atributo identificador da nave na Missão, e também adicionar um marcador de não nulo (“*”).

Vantagens:

Método adotado apresenta relação direta entre cada tipo de nave com sua missão específica. Há garantia de que não haverá

- **Relacionamento N-N Missão de Exploração explora Recurso Natural:/**

Solução Adotada: Criou-se uma nova tabela que contém o campo identificador da missão de exploração, os campos que identifica o recurso natural, e o atributo quantidade que é referente a quantidade de determinado recurso naquela missão de exploração. Essa nova tabela possui como identificador as chaves referentes às entidades Missão de Exploração e Recursos Naturais.

Vantagens:

O modelo adotado garante que haja a possibilidade relacionar vários recursos em uma mesma missão de exploração, como também que um recurso esteja associado a várias missões de exploração.

Ao mapear esse relacionamento criando uma nova tabela possibilita uma maior facilidade de consulta e acesso em informações referente a entidades associadas.

O Emprego da chave primária ser as chaves das entidades envolvidas faz com que garanta maior consistência, fazendo com que não tenha repetição do relacionamento entre uma missão de exploração com determinado recurso.

Para esse caso em que há um atributo de relacionamento e possibilidade de vários relacionamentos entre entidades (N:N) não há outra solução alternativa eficaz.

- **Relacionamento 1-N Missão de Colonização coloniza Planeta:**

Solução Adotada: Adicionou-se o atributo identificador do Planeta na tabela de Missão de Colonização, e também acrescentou um marcador de não nulo para o campo acrescentado.

Vantagens:

Faz que toda missão colonizadora esteja associada a um planeta e que sua participação total esteja sendo garantida.

Permite maior facilidade de consultas relacionadas a missões em um determinado planeta. Também possui uma compreensão mais trivial do relacionamento entre as entidades.

- **Relacionamento N:N Civilização é ameaça:**

Solução Adotada: Para o mapeamento desse relacionamento criou-se uma nova tabela contendo o identificador da civilização e também o identificador da ameaça. Utilizou-se os campos de atributos adicionados como campo identificador da tabela criada.

Vantagens:

Ao utilizar chaves estrangeiras como identificador único há uma garantia de não repetição de informações. No caso, uma civilização não pode ser relacionada mais de uma vez com o mesmo tipo de ameaça.

O modelo também garante uma multiplicidade dos relacionamentos, possibilitando que uma civilização possa ser vários tipos de ameaças.

Desvantagens:

Não há garantia de que a civilização que vai se relacionar é apenas as que são Amistosa, ou seja, não garante que apenas civilizações Amistosas possam ser consideradas ameaças.

- **Relacionamento N-N-N Missão de Comércio comercializa Produto com Civilização:**

Solução Adotada: Criou-se uma tabela para mapear o relacionamento, a qual possui como campos a missão, o nome do produto, origem do produto, civilização, tipo comércio e quantidade. Os atributos tipo de comércio e quantidade são advindos do relacionamento das três entidades envolvidas (Missão comércio, produto e civilização). Utilizou-se como identificador dessa tabela criada os campos de identificação da Missão, os campos de identificação do produto e os campos de identificação da civilização.

Vantagens:

Garante normalização, que ajuda a reduzir a duplicação de dados e manter sua consistência. Forma mais intuitiva de visualizar relacionamentos considerados mais complexos, no caso um relacionamento que envolve mais de duas entidades, deixando mais explícita as entidades relacionadas e suas ligações. Mapeando dessa forma também irá garantir uma maior facilidade de busca em entidades relacionadas, dado que seus relacionamentos são apresentados de forma estruturada.

Uma das principais vantagens é a adição do atributo referente ao relacionamento da Missão comércio, produto e civilização, pois a quantidade é de um dado produto em uma

missão específica, não faria sentido colocar em nenhuma das outras entidades específicas. O mesmo se aplica para o atributo de tipo de comércio.

Desvantagem:

Garantia de existência do produto, missão de comércio e civilização antes de qualquer inserção na tabela

Para esse caso em que há atributos de relacionamento e possibilidade de vários relacionamentos entre entidades (N:N:N) não há outra solução alternativa eficaz.

9.1. Soluções alternativas:

9.1.1. Para Mapeamento 1:N

Em todos os relacionamentos 1:N adotamos a solução de adicionar o identificador da entidade do lado “1”, da cardinalidade, na tabela da entidade do lado “N”. Esse método foi adotado pelo fato de que todos esses relacionamentos, no projeto, garantia uma participação muito grande da entidade do lado “1” com a do lado “N”.

A solução alternativa para esse tipo de relacionamento é criar um relacionamento do tipo 1:1, que é dado a partir da criação de uma tabela onde é feito uma mescla dos atributos de ambas as tabelas;

No caso, essa solução não é muito aplicável a esse relacionamento por sua limitação de que sempre ao buscar uma das entidades resultará também campos associados a outra entidade, podendo estes estarem preenchidos ou nulos, gerando uma complexidade maior no resultado obtido das consultas. E também é possível obter muitos campos nulos. Como por exemplo no caso da Base e Nave, em que pode haver Bases que não construíram nenhuma nave.

Essa mescla só iria fazer sentido em um caso em que as duas entidades compartilham a maioria dos atributos, que não é o que ocorre com nenhuma das entidades.

9.1.2. Para atributos Multivalorados

Para todos os casos de atributos multivalorados criou-se uma nova tabela, a qual era composta pela chave da tabela de origem e o valor do campo do atributo, sendo também esses atributos seu identificador único, para assim garantir a não repetição das informações.

Uma outra alternativa seria mapear os atributos multivalorados na sua tabela de origem. Por exemplo, no caso das Estrelas criar os campos Estrela1, Estrela2 e Estrela3.

No entanto, a adoção desse método implicaria em uma alta probabilidade de duplicação de dados, o que iria acabar ocupando mais espaço de armazenamento e aumentando a complexidade na atualização e manutenção dos dados.

Outros pontos negativos também poderiam ser resultados. Como, por exemplo, a geração de espaços nulos caso não fosse fixo o número de inserções, a pouca flexibilização do número de campos a serem inseridos.

A adição na tabela de origem só faz sentido quando tem uma certa garantia de número de atributos envolvidos. Como é o caso de telefones, em que é muito difícil encontrar alguém que possua mais de três números de telefone;

9.1.3. Para Mapeamento N:N (ou N:N:N)

Não há soluções alternativas para mapeamentos do tipo muitos para muitos.

10. Previsto no MER e que não foi possível garantir no Relacional:

- Não é possível garantir a participação total no relacionamento ternário Civilização comercializar Produto na Missão de Comércio.
- Não foi possível garantir a especialização total das Naves e das Missões Especiais, pois não há como garantir que as entidades serão cadastradas nas tabelas do conjunto de entidades específicas.
- Não é possível garantir participação total do relacionamento N:N entre Planeta e Recurso Natural. (Todo Recurso está Associado a um planeta).

11. Criação da Base de Dados:

Para algumas tabelas foi utilizada a cláusula específica ON DELETE CASCADE, em uma restrição de chave estrangeira indica que, se uma linha referenciada na tabela pai (referenciada) for excluída, todas as linhas correspondentes nas tabelas filhas (referenciadoras) também serão automaticamente excluídas.

```
CREATE TABLE SISTEMA_PLANETARIO(
    NOME VARCHAR2(50),
    GALAXIA VARCHAR2(50),
    DISTANCIA NUMBER(20),
    IDADE_ESTIMADA NUMBER(20),
    TIPO_ESTRELA_CENTRAL VARCHAR2(20) NOT NULL,
    CONSTRAINT PK_SISTEMA_PLANETARIO PRIMARY KEY (NOME, GALAXIA)
);
```

A tabela foi projetada para armazenar informações sobre sistemas planetários, incluindo detalhes como nome, galáxia, distância, idade estimada e tipo de estrela central. A constraint PK_SISTEMA_PLANETARIO é uma restrição de chave primária composta pelas colunas NOME e GALAXIA, garantindo unicidade na combinação desses dois campos.

```
CREATE TABLE ESTRELA(
    NOME_SISTEMA VARCHAR2(50),
    GALAXIA VARCHAR2(50),
    ESTRELA VARCHAR2(50),
    CONSTRAINT PK_ESTRELA PRIMARY KEY (NOME_SISTEMA, GALAXIA,
ESTRELA),
    CONSTRAINT FK_ESTRELA FOREIGN KEY (NOME_SISTEMA, GALAXIA)
REFERENCES SISTEMA_PLANETARIO(NOME, GALAXIA) ON DELETE CASCADE
);
```

A tabela ESTRELA é projetada para armazenar informações sobre estrelas dentro de sistemas planetários, mantendo uma integridade referencial com a tabela “SISTEMA_PLANETARIO” e permitindo a exclusão automática de entradas relacionadas quando o sistema planetário correspondente é excluído. A FK_ESTRELA é uma restrição de chave estrangeira que referencia as colunas “NOME”.

```
CREATE TABLE PLANETA(
    ID NUMBER(15),
    SISTEMA_PLANETARIO VARCHAR2(50) NOT NULL,
    GALAXIA VARCHAR2(50) NOT NULL,
    NOME VARCHAR2(50) NOT NULL,
    TEMPERATURA FLOAT,
    PRESSAO FLOAT,
    CLIMA VARCHAR2(20),
    CONSTRAINT PK_PLANETA PRIMARY KEY (ID),
```

```

        CONSTRAINT FK_PLANETA FOREIGN KEY (SISTEMA_PLANETARIO,
        GALAXIA) REFERENCES SISTEMA_PLANETARIO(NOME, GALAXIA) ON DELETE
        CASCADE,
        CONSTRAINT SK_PLANETA UNIQUE (SISTEMA_PLANETARIO, GALAXIA,
        NOME)
    );

```

A tabela PLANETA foi projetada para armazenar informações sobre planetas, incluindo detalhes como identificação, sistema planetário, galáxia, nome, temperatura, pressão e clima, mantendo integridade referencial com a tabela SISTEMA_PLANETARIO. A FK_PLANETA é uma restrição de chave estrangeira que referencia as colunas SISTEMA_PLANETARIO e GALAXIA da tabela SISTEMA_PLANETARIO

```

CREATE TABLE SATELITE_NATURAL(
    PLANETA NUMBER(15),
    SATELITE VARCHAR2(50),
    CONSTRAINT PK_SATELITE_NATURAL PRIMARY KEY (PLANETA, SATELITE),
    CONSTRAINT FK_SATELITE_NATURAL FOREIGN KEY (PLANETA)
    REFERENCES PLANETA(ID) ON DELETE CASCADE
);

```

A tabela SATELITE_NATURAL foi criada para armazenar informações sobre satélites naturais associados a planetas, garantindo integridade referencial com a tabela PLANETA e permitindo a exclusão automática de satélites naturais quando o planeta correspondente é excluído.

```

CREATE TABLE COMPOSICAO_ATMOSFERICA(
    PLANETA NUMBER(15),
    ELEMENTO VARCHAR2(50),
    PORCENTAGEM NUMBER(3) NOT NULL,
    CONSTRAINT PK_COMPOSICAO_ATMOSFERICA PRIMARY KEY (PLANETA,
    ELEMENTO),
    CONSTRAINT FK_COMPOSICAO_ATMOSFERICA FOREIGN KEY (PLANETA)
    REFERENCES PLANETA(ID) ON DELETE CASCADE
);

```

A tabela COMPOSICAO_ATMOSFERICA foi concebida para armazenar dados referentes à composição atmosférica de planetas. Essa estrutura assegura a integridade referencial, mantendo uma conexão válida com a tabela PLANETA. Além disso, está configurada para possibilitar a exclusão automática das entradas correspondentes na tabela COMPOSICAO_ATMOSFERICA no caso de exclusão do planeta associado, garantindo consistência nos dados.

```

CREATE TABLE CIVILIZACAO(
    NOME VARCHAR2(50),
    CATEGORIA CHAR(1),
    CLASSIFICACAO CHAR(3) NOT NULL,
    PLANETA_ORIGEM NUMBER(15) NOT NULL,
    CONSTRAINT PK_CIVILIZACAO PRIMARY KEY (NOME, CATEGORIA),

```

```

        CONSTRAINT FK_CIVILIZACAO FOREIGN KEY (PLANETA_ORIGEM)
REFERENCES PLANETA(ID) ON DELETE CASCADE
);

```

A tabela CIVILIZACAO foi criada para armazenar dados relativos a civilizações, abrangendo informações como nome, categoria, classificação e o planeta de origem associado. Essa estrutura é concebida para garantir a integridade referencial com a tabela PLANETA, mantendo uma conexão válida. Além disso, está configurada para permitir a exclusão automática de entradas correspondentes na tabela CIVILIZACAO no caso de exclusão do planeta de origem associado, assegurando consistência nos dados.

```

CREATE TABLE CIVILIZACAO_HABITA_PLANETA(
    NOME_CIVILIZACAO VARCHAR2(25),
    CATEGORIA_CIVILIZACAO CHAR(1),
    PLANETA NUMBER(15),
    CONSTRAINT PK_CIVILIZACAO_HABITA_PLANETA PRIMARY KEY
(NOME_CIVILIZACAO, CATEGORIA_CIVILIZACAO, PLANETA),
    CONSTRAINT FK_CIVILIZACAO_HABITA_PLANETA1 FOREIGN KEY
(NOME_CIVILIZACAO, CATEGORIA_CIVILIZACAO) REFERENCES CIVILIZACAO(NOME,
CATEGORIA) ON DELETE CASCADE,
    CONSTRAINT FK_CIVILIZACAO_HABITA_PLANETA2 FOREIGN KEY
(PLANETA) REFERENCES PLANETA(ID) ON DELETE CASCADE
);

```

Essa tabela foi projetada para armazenar informações sobre a relação entre civilizações e os planetas que habitam, mantendo integridade referencial com as tabelas CIVILIZACAO e PLANETA. A FK_CIVILIZACAO_HABITA_PLANETA1 é uma restrição de chave estrangeira que faz referência às colunas NOME_CIVILIZACAO e CATEGORIA_CIVILIZACAO da tabela CIVILIZACAO. Se uma civilização associada a um planeta for excluída, as informações correspondentes na tabela CIVILIZACAO_HABITA_PLANETA também serão eliminadas. A FK_CIVILIZACAO_HABITA_PLANETA2 é uma restrição de chave estrangeira que faz referência à coluna PLANETA da tabela PLANETA. Garante que se um planeta habitado por uma civilização for excluído, as informações correspondentes na tabela CIVILIZACAO_HABITA_PLANETA também serão excluídas.

```

CREATE TABLE ROTA(
    ORIGEM NUMBER(15),
    DESTINO NUMBER(15),
    DISTANCIA NUMBER(20) NOT NULL,
    CONSTRAINT PK_ROTA PRIMARY KEY (ORIGEM, DESTINO),
    CONSTRAINT FK_ROTA1 FOREIGN KEY (ORIGEM) REFERENCES
PLANETA(ID) ON DELETE CASCADE,
    CONSTRAINT FK_ROTA2 FOREIGN KEY (DESTINO) REFERENCES
PLANETA(ID) ON DELETE CASCADE,
    CONSTRAINT CK_ROTA CHECK (ORIGEM <> DESTINO)
);

```

A tabela ROTA foi projetada para armazenar informações sobre rotas entre planetas, mantendo integridade referencial com a tabela PLANETA e possibilitando a exclusão automática de registros quando as entidades relacionadas são removidas. A restrição de verificação (CK_ROTAS) impede que a origem e o destino sejam idênticos. A FK_ROTAS1 é uma restrição de chave estrangeira que faz referência à coluna ORIGEM da tabela PLANETA, garantindo que se um planeta de origem associado a uma rota for excluído, a própria rota também será eliminada. Já a FK_ROTAS2 é uma restrição de chave estrangeira que faz referência à coluna DESTINO da tabela PLANETA, utilizando a opção CASCADE, o que significa que se um planeta de destino associado a uma rota for excluído, a rota também será excluída automaticamente.

```
CREATE TABLE AMEACA (
    NOME VARCHAR2(50),
    DESCRICAO VARCHAR2(100) NOT NULL,
    NIVEL NUMBER(5) NOT NULL,
    CONSTRAINT PK_AMEACA PRIMARY KEY (NOME) ON DELETE
    CASCADE
);
```

A tabela AMEACA foi criada para armazenar informações sobre ameaças, onde o nome serve como chave primária e a descrição detalha a natureza da ameaça. O nível representa a gravidade da ameaça, e a configuração ON DELETE CASCADE na chave primária permite a exclusão automática de referências a uma ameaça quando ela é removida.

```
CREATE TABLE BASE_ESPACIAL (
    PLANETA NUMBER(15),
    NOME VARCHAR2(50),
    TIPO VARCHAR2(25),
    CAPACIDADE NUMBER(10),
    CANAL_COMUNICACAO VARCHAR2(50),
    CONSTRAINT PK_BASE_ESPACIAL PRIMARY KEY (NOME, PLANETA),
    CONSTRAINT FK_BASE_ESPACIAL FOREIGN KEY (PLANETA)
    REFERENCES PLANETA(ID) ON DELETE CASCADE
);
```

A tabela BASE_ESPACIAL foi feita para armazenar informações sobre bases espaciais, associando-as a planetas específicos. A chave primária composta (NOME e PLANETA) garante a unicidade de cada base espacial, e a chave estrangeira (FK_BASE_ESPACIAL) mantém a integridade referencial com a tabela PLANETA, permitindo a exclusão automática de bases espaciais quando os planetas associados são removidos.

```
CREATE TABLE MISSAO_ESPACIAL (
    ID NUMBER(15),
    PLANETA_BASE NUMBER(15) NOT NULL,
    NOME_BASE VARCHAR2(50) NOT NULL,
```

```

NOME VARCHAR2(50) NOT NULL,
DATA_INICIO DATE NOT NULL,
DATA_FIM DATE NOT NULL,
DESCRICAO VARCHAR2(100) NOT NULL,
TAMANHO_TRIPULACAO NUMBER(15),
OBJETIVO VARCHAR2(50) NOT NULL,
DURACAO_ESTIMADA NUMBER(15),
NIVEL_PERICULOSIDADE NUMBER(15) NOT NULL,
    CONSTRAINT PK_MISSAO_ESPACIAL PRIMARY KEY (ID),
    CONSTRAINT FK_MISSAO_ESPACIAL FOREIGN KEY (PLANETA_BASE,
NOME_BASE) REFERENCES BASE_ESPACIAL(PLANETA, NOME) ON DELETE
CASCADE,
    CONSTRAINT SK_MISSAO_ESPACIAL UNIQUE (PLANETA_BASE,
NOME_BASE, NOME, DATA_INICIO),
    CONSTRAINT CK_MISSAO_ESPACIAL CHECK (DATA_FIM > DATA_INICIO)
);

```

A tabela `MISSAO_ESPACIAL` foi projetada para armazenar informações específicas sobre missões espaciais. Ela incorpora diversas restrições para garantir a integridade e consistência dos dados. Uma dessas restrições é a `FK_MISSAO_ESPACIAL`, uma chave estrangeira que se refere às colunas `PLANETA_BASE` e `NOME_BASE` da tabela `BASE_ESPACIAL`. Essa configuração permite a exclusão em cascata (`ON DELETE CASCADE`), o que significa que se a base espacial associada a uma missão for removida, a própria missão será automaticamente excluída. Além disso, a `SK_MISSAO_ESPACIAL` é uma restrição de chave única nas colunas `PLANETA_BASE`, `NOME_BASE`, `NOME` e `DATA_INICIO`, assegurando a singularidade dessa combinação de atributos. A `CK_MISSAO_ESPACIAL` é uma restrição de verificação que valida se a data de término (`DATA_FIM`) é posterior à data de início (`DATA_INICIO`). Isso garante que as datas estejam sempre ordenadas de maneira lógica. Essas restrições são implementadas para manter a consistência e a integridade dos dados na tabela `MISSAO_ESPACIAL`. A `FK_MISSAO_ESPACIAL` preserva a relação apropriada entre missões e bases espaciais, a `SK_MISSAO_ESPACIAL` evita a duplicação de missões com os mesmos atributos-chave, e a `CK_MISSAO_ESPACIAL` assegura a ordenação lógica das datas. A exclusão em cascata facilita a manutenção da integridade referencial ao automatizar a remoção de missões quando as bases espaciais associadas são excluídas.

```

CREATE TABLE NAVE (
    ID NUMBER(15),
    NOME VARCHAR2(50) NOT NULL,
    CLASSE VARCHAR2(50) NOT NULL,
    MAX_TRIPULACAO NUMBER(15),
    RESISTENCIA NUMBER(15),
    LOC_ATUAL VARCHAR2(50) NOT NULL,
    PLANETA_BASE NUMBER(15) NOT NULL,
    NOME_BASE VARCHAR2(50) NOT NULL,
    CONSTRAINT PK_NAVE PRIMARY KEY (ID),

```

```

        CONSTRAINT FK_NAVE FOREIGN KEY (PLANETA_BASE, NOME_BASE)
REFERENCES BASE_ESPACIAL(PLANETA, NOME),
        CONSTRAINT SK_NAVE UNIQUE (NOME, CLASSE)
);

```

A tabela NAVE foi criada para fornecer informações detalhadas sobre naves espaciais, incluindo seu nome, classe, capacidade de tripulação, resistência, localização atual e a relação com a base espacial associada. A integridade dos dados é mantida por meio de restrições de chave primária, chave estrangeira e chave única. A SK_NAVE é uma restrição de chave única definida nas colunas NOME e CLASSE, garantindo a unicidade dessa combinação de atributos. A FK_NAVE é uma restrição de chave estrangeira que referencia as colunas PLANETA_BASE e NOME_BASE da tabela BASE_ESPACIAL, essa restrição estabelece uma relação entre a nave e a base espacial à qual ela está associada.

```

CREATE TABLE AMEACA_ROTA(
        ORIGEM_ROTA NUMBER(15),
        DESTINO_ROTA NUMBER(15),
        NOME_AMEACA VARCHAR2(50),
        CONSTRAINT PK_AMEACA_ROTA PRIMARY KEY (ORIGEM_ROTA,
DESTINO_ROTA, NOME_AMEACA),
        CONSTRAINT FK_AMEACA_ROTA1 FOREIGN KEY (ORIGEM_ROTA,
DESTINO_ROTA) REFERENCES ROTA(ORIGEM, DESTINO) ON DELETE CASCADE,
        CONSTRAINT FK_AMEACA_ROTA2 FOREIGN KEY (NOME_AMEACA)
REFERENCES AMEACA(NOME) ON DELETE CASCADE
);

```

Essa estrutura permite registrar e manter a integridade referencial das ameaças relacionadas às rotas, garantindo que as informações permaneçam consistentes mesmo com alterações nas tabelas relacionadas.

```

CREATE TABLE ROTA_MISSAO(
        MISSAO NUMBER(15),
        ORIGEM_ROTA NUMBER(15),
        DESTINO_ROTA NUMBER(15),
        ORDEM NUMBER(15) NOT NULL,
        CONSTRAINT PK_ROTAMISSAO PRIMARY KEY (MISSAO,
ORIGEM_ROTA, DESTINO_ROTA),
        CONSTRAINT FK_ROTAMISSAO1 FOREIGN KEY (MISSAO)
REFERENCES MISSAO_ESPACIAL(ID) ON DELETE CASCADE,
        CONSTRAINT FK_ROTAMISSAO2 FOREIGN KEY (ORIGEM_ROTA,
DESTINO_ROTA) REFERENCES ROTA(ORIGEM, DESTINO) ON DELETE CASCADE,
        CONSTRAINT SK_ROTAMISSAO UNIQUE(MISSAO, ORDEM)
);

```

A tabela ROTA_MISSAO foi criada para registrar informações sobre a relação entre missões espaciais e as rotas associadas a elas. A estrutura possibilita a associação ordenada de rotas a missões espaciais, mantendo a consistência dos dados e permitindo a remoção automática de registros associados quando missões ou rotas são excluídas. A FK_ROTAMISSAO1 é chave Estrangeira na coluna MISSAO, referenciando a tabela MISSAO_ESPACIAL e configurada para exclusão em cascata, implica que se a missão

associada à rota for excluída, a própria entrada na tabela ROTA_MISSAO também será excluída. A FK_ROTAMISSAO2 é chave estrangeira nas colunas ORIGEM_ROTAMISSAO e DESTINO_ROTAMISSAO, referenciando a tabela ROTA e foi configurada com ON DELETE CASCADE, e assim quando rota associada à missão for excluída, a própria entrada na tabela ROTA_MISSAO também será excluída. A SK_ROTAMISSAO é a chave única nas colunas MISSAO e ORDEM, garantindo a unicidade dessa combinação de atributos.

```
CREATE TABLE AMEACA_PLANETA (
    PLANETA NUMBER(15),
    AMEACA VARCHAR2(50),
    CONSTRAINT PK_AMEACA_PLANETA PRIMARY KEY (PLANETA,
AMEACA),
    CONSTRAINT FK_AMEACA_PLANETA1 FOREIGN KEY (PLANETA)
REFERENCES PLANETA(ID) ON DELETE CASCADE,
    CONSTRAINT FK_AMEACA_PLANETA2 FOREIGN KEY (AMEACA)
REFERENCES AMEACA(NOME) ON DELETE CASCADE
);
```

Essa tabela é destinada a armazenar informações sobre as ameaças relacionadas aos planetas. Ela possui duas colunas: PLANETA para armazenar o identificador do planeta e AMEACA para armazenar o nome da ameaça ao planeta. Adicionalmente, são definidas três restrições, sendo estas a chave primária da tabela, chamada PK_AMEACA_PLANETA, é estabelecida como a combinação das colunas PLANETA e AMEACA, assegurando a unicidade dos registros. Uma restrição de chave estrangeira, FK_AMEACA_PLANETA1, vincula a coluna PLANETA na tabela AMEACA_PLANETA à coluna ID na tabela PLANETA. Isso implica que o valor em PLANETA deve corresponder a um valor existente na coluna ID da tabela PLANETA. A cláusula ON DELETE CASCADE indica que, se um registro na tabela PLANETA for excluído, os registros correspondentes na tabela AMEACA_PLANETA também serão removidos. Outra restrição de chave estrangeira, FK_AMEACA_PLANETA2, vincula a coluna AMEACA na tabela AMEACA_PLANETA à coluna NOME na tabela AMEACA. Similar à primeira restrição, isso implica que o valor em AMEACA deve corresponder a um valor existente na coluna NOME da tabela AMEACA. A cláusula ON DELETE CASCADE indica que, se um registro na tabela AMEACA for excluído, os registros correspondentes na tabela AMEACA_PLANETA também serão excluídos.

```
CREATE TABLE AMEACA_CIVILIZACAO (
    AMEACA VARCHAR2(25),
    NOME_CIV VARCHAR2(25),
    CATEG_CIV CHAR(1),
    CONSTRAINT PK_AMEACA_CIVILIZACAO PRIMARY KEY (AMEACA,
NOME_CIV, CATEG_CIV),
    CONSTRAINT FK_AMEACA_CIVILIZACAO1 FOREIGN KEY (AMEACA)
REFERENCES AMEACA(NOME) ON DELETE CASCADE,
    CONSTRAINT FK_AMEACA_CIVILIZACAO2 FOREIGN KEY (NOME_CIV,
CATEG_CIV) REFERENCES CIVILIZACAO(NOME, CATEGORIA) ON DELETE CASCADE
```


);

A tabela foi feita para armazenar informações sobre ameaças relacionadas a civilizações. A tabela possui três colunas: AMEACA para armazenar o nome da ameaça, NOME_CIV para armazenar o nome da civilização, e CATEG_CIV para armazenar a categoria da civilização. A PK_AMEACA_CIVILIZACAO estabelece a chave primária da tabela como sendo a combinação das colunas AMEACA, NOME_CIV, e CATEG_CIV, garantindo unicidade de registros. A FK_AMEACA_CIVILIZACAO1 é uma restrição de chave estrangeira que vincula a coluna AMEACA na tabela AMEACA_CIVILIZACAO à coluna NOME na tabela AMEACA. Isso implica que o valor em AMEACA deve corresponder a um valor existente na coluna NOME da tabela AMEACA. A cláusula ON DELETE CASCADE indica que, se um registro na tabela AMEACA for excluído, os registros correspondentes na tabela AMEACA_CIVILIZACAO também serão removidos. A FK_AMEACA_CIVILIZACAO2 é outra restrição de chave estrangeira vincula as colunas NOME_CIV e CATEG_CIV na tabela AMEACA_CIVILIZACAO às colunas NOME e CATEGORIA na tabela CIVILIZACAO. Isso implica que a combinação de valores em NOME_CIV e CATEG_CIV deve corresponder a uma combinação existente na tabela CIVILIZACAO. A cláusula ON DELETE CASCADE indica que, se um registro na tabela CIVILIZACAO for excluído, os registros correspondentes na tabela AMEACA_CIVILIZACAO também serão excluídos.

```
CREATE TABLE RECURSO_NATURAL (
    NOME VARCHAR2(50),
    TIPO VARCHAR2(50),
    CONSTRAINT PK_RECURSO_NATURAL PRIMARY KEY (NOME, TIPO)
);
```

A tabela RECURSO_NATURAL armazena informações sobre recursos naturais, sendo caracterizada por duas colunas principais que possibilita a identificação única de recursos naturais com base em seus nomes e tipos, oferecendo uma maneira eficiente de gerenciar e consultar informações sobre diversos recursos presentes no contexto da aplicação.

```
CREATE TABLE PRODUTO (
    NOME VARCHAR2(50),
    ORIGEM VARCHAR2(50),
    DESCRICAO VARCHAR2(100) NOT NULL,
    PRECO NUMBER(15,2) NOT NULL,
    PESO NUMBER(15,2),
    CONSTRAINT PK_PRODUTO PRIMARY KEY (NOME, ORIGEM)
);
```

A tabela foi criada para armazenar informações sobre produtos, possui colunas como nome, origem, descrição, preço, peso, e também há uma restrição chamada PK_PRODUTO que define uma chave primária composta pelas colunas "NOME" e "ORIGEM". Essa restrição garante que a combinação única desses valores identifica cada registro de forma exclusiva na tabela, impedindo duplicatas na chave primária.

```

CREATE TABLE RECURSO_PLANETA(
    PLANETA NUMBER(15),
    NOME_REC VARCHAR2(50),
    tipo_REC VARCHAR2(50),
    ESTIMATIVA NUMBER(15),
    CONSTRAINT PK_RECURSO_PLANETA PRIMARY KEY (PLANETA,
    TIPO_REC, NOME_REC),
    CONSTRAINT FK_RECURSO_PLANETA1 FOREIGN KEY (PLANETA)
REFERENCES PLANETA(ID) ON DELETE CASCADE,
    CONSTRAINT FK_RECURSO_PLANETA2 FOREIGN KEY (NOME_REC,
    TIPO_REC) REFERENCES RECURSO_NATURAL(NOME, TIPO) ON DELETE CASCADE
);

```

A tabela RECURSO_PLANETA é destinada a armazenar informações sobre recursos naturais em planetas. A tabela possui quatro colunas: PLANETA para armazenar o identificador do planeta, NOME_REC para armazenar o nome do recurso, TIPO_REC para armazenar o tipo do recurso, e ESTIMATIVA para armazenar uma estimativa relacionada ao recurso. A PK_RECURSO_PLANETA estabelece a chave primária da tabela como sendo a combinação das colunas PLANETA, TIPO_REC, e NOME_REC, assegurando a unicidade dos registros. A FK_RECURSO_PLANETA1 é uma restrição de chave estrangeira que vincula a coluna PLANETA na tabela RECURSO_PLANETA à coluna ID na tabela PLANETA. Isso implica que o valor em PLANETA deve corresponder a um valor existente na coluna ID da tabela PLANETA. A cláusula ON DELETE CASCADE indica que, se um registro na tabela PLANETA for excluído, os registros correspondentes na tabela RECURSO_PLANETA também serão removidos. A FK_RECURSO_PLANETA2 é outra restrição de chave estrangeira vincula as colunas NOME_REC e TIPO_REC na tabela RECURSO_PLANETA às colunas NOME e TIPO na tabela RECURSO_NATURAL. Isso implica que a combinação de valores em NOME_REC e TIPO_REC deve corresponder a uma combinação existente na tabela RECURSO_NATURAL. A cláusula ON DELETE CASCADE indica que, se um registro na tabela RECURSO_NATURAL for excluído, os registros correspondentes na tabela RECURSO_PLANETA também serão excluídos.

```

CREATE TABLE NAVE_COLONIZADORA (
    NAVE NUMBER(15),
    CONSTRAINT PK_NAVE_COLONIZADORA PRIMARY KEY (NAVE),
    CONSTRAINT FK_NAVE_COLONIZADORA FOREIGN KEY (NAVE)
REFERENCES NAVE(ID) ON DELETE CASCADE
);

```

Esta tabela foi feita para armazenar informações sobre naves colonizadoras e possui apenas uma coluna: NAVE para armazenar o identificador da nave colonizadora. A PK_NAVE_COLONIZADORA estabelece a chave primária da tabela como sendo a coluna NAVE, garantindo a unicidade dos registros. A FK_NAVE_COLONIZADORA é uma restrição de chave estrangeira que vincula a coluna NAVE na tabela NAVE_COLONIZADORA à coluna ID na tabela NAVE. Isso implica que o valor em NAVE deve corresponder a um valor existente na coluna ID da tabela NAVE. A cláusula ON DELETE CASCADE indica que, se um registro na tabela NAVE for excluído, os registros correspondentes na tabela NAVE_COLONIZADORA também serão removidos.

```

CREATE TABLE NAVE_CARGA(
    NAVE NUMBER(15),
    CAPACIDADE_CARGA NUMBER(15) NOT NULL,
    CONSTRAINT PK_NAVE_CARGA PRIMARY KEY (NAVE),
    CONSTRAINT FK_NAVE_CARGA FOREIGN KEY (NAVE) REFERENCES
NAVE(ID) ON DELETE CASCADE
);

```

Essa tabela é destinada a armazenar informações sobre naves de carga e possui duas colunas: NAVE para armazenar o identificador da nave e CAPACIDADE_CARGA para armazenar a capacidade de carga da nave, a qual não pode ser nula. A PK_NAVE_CARGA estabelece a chave primária da tabela como sendo a coluna NAVE, garantindo a unicidade dos registros. A FK_NAVE_CARGA é uma restrição de chave estrangeira que vincula a coluna NAVE na tabela NAVE_CARGA à coluna ID na tabela NAVE. Isso implica que o valor em NAVE deve corresponder a um valor existente na coluna ID da tabela NAVE. A cláusula ON DELETE CASCADE indica que, se um registro na tabela NAVE for excluído, os registros correspondentes na tabela NAVE_CARGA também serão removidos.

```

CREATE TABLE NAVE_CIENTIFICA(
    NAVE NUMBER(15),
    CAPACIDADE_CARGA NUMBER(15) NOT NULL,
    CONSTRAINT PK_NAVE_CIENTIFICA PRIMARY KEY (NAVE),
    CONSTRAINT FK_NAVE_CIENTIFICA FOREIGN KEY (NAVE)
REFERENCES NAVE(ID) ON DELETE CASCADE
);

```

A tabela NAVE_CIENTIFICA é projetada para armazenar informações sobre naves científicas e possui duas colunas: NAVE para armazenar o identificador da nave e CAPACIDADE_CARGA para representar a capacidade de carga da nave, a qual não pode ser nula. A PK_NAVE_CIENTIFICA:** Estabelece a chave primária da tabela como sendo a coluna NAVE, garantindo a unicidade dos registros. A FK_NAVE_CIENTIFICA:** Uma restrição de chave estrangeira que vincula a coluna NAVE na tabela NAVE_CIENTIFICA à coluna ID na tabela NAVE. Isso implica que o valor em NAVE deve corresponder a um valor existente na coluna ID da tabela NAVE. A cláusula ON DELETE CASCADE indica que, se um registro na tabela NAVE for excluído, os registros correspondentes na tabela NAVE_CIENTIFICA também serão removidos.

```

CREATE TABLE PRODUTO_CIVILIZACAO (
    NOME_CIV VARCHAR2(50),
    CATEG_CIV CHAR(1),
    NOME_PRODUTO VARCHAR2(50),
    ORIGEM_PRODUTO VARCHAR2(50),
    CONSTRAINT PK_ PRIMARY KEY (NOME_CIV, CATEG_CIV,
NOME_PRODUTO, ORIGEM_PRODUTO),
    CONSTRAINT FK_PRODUTO_CIVILIZACAO1 FOREIGN KEY
(NOME_PRODUTO, ORIGEM_PRODUTO) REFERENCES PRODUTO(NOME, ORIGEM)
ON DELETE CASCADE,

```

```

        CONSTRAINT FK_PRODUTO_CIVILIZACAO2 FOREIGN KEY (NOME_CIV,
CATEG_CIV) REFERENCES CIVILIZACAO(NOME, CATEGORIA) ON DELETE CASCADE
);

```

A tabela PRODUTO_CIVILIZACAO é destinada a armazenar informações sobre a relação entre produtos e civilizações. A tabela possui quatro colunas: NOME_CIV para armazenar o nome da civilização, CATEG_CIV para armazenar a categoria da civilização, NOME_PRODUTO para armazenar o nome do produto, e ORIGEM_PRODUTO para armazenar a origem do produto. A PK_ estabelece a chave primária da tabela como sendo a combinação das colunas NOME_CIV, CATEG_CIV, NOME_PRODUTO, e ORIGEM_PRODUTO, garantindo a unicidade dos registros. A FK_PRODUTO_CIVILIZACAO1 uma restrição de chave estrangeira que vincula as colunas NOME_PRODUTO e ORIGEM_PRODUTO na tabela PRODUTO_CIVILIZACAO às colunas NOME e ORIGEM na tabela PRODUTO. Isso implica que a combinação de valores em NOME_PRODUTO e ORIGEM_PRODUTO deve corresponder a uma combinação existente na tabela PRODUTO. A cláusula ON DELETE CASCADE indica que, se um registro na tabela PRODUTO for excluído, os registros correspondentes na tabela PRODUTO_CIVILIZACAO também serão removidos. A FK_PRODUTO_CIVILIZACAO2 outra restrição de chave estrangeira vincula as colunas NOME_CIV e CATEG_CIV na tabela PRODUTO_CIVILIZACAO às colunas NOME e CATEGORIA na tabela CIVILIZACAO. Isso implica que a combinação de valores em NOME_CIV e CATEG_CIV deve corresponder a uma combinação existente na tabela CIVILIZACAO. A cláusula ON DELETE CASCADE indica que, se um registro na tabela CIVILIZACAO for excluído, os registros correspondentes na tabela PRODUTO_CIVILIZACAO também serão excluídos.

```

CREATE TABLE MISSAO_EXPLORACAO (
    MISSAO NUMBER(15),
    NIVEL_IMPACTO_AMBIENTAL NUMBER(15),
    NAVE NUMBER(15) NOT NULL,
    PLANETA_BASE_DEST NUMBER(15) NOT NULL,
    NOME_BASE_DEST VARCHAR2(50) NOT NULL,
    CONSTRAINT PK_MISSAO_EXPLORACAO PRIMARY KEY (MISSAO),
    CONSTRAINT FK_MISSAO_EXPLORACAO1 FOREIGN KEY (MISSAO)
REFERENCES MISSAO_ESPACIAL(ID) ON DELETE CASCADE,
    CONSTRAINT FK_MISSAO_EXPLORACAO2 FOREIGN KEY (NAVE)
REFERENCES NAVE(ID) ON DELETE CASCADE,
    CONSTRAINT FK_MISSAO_EXPLORACAO3 FOREIGN KEY
(PLANETA_BASE_DEST, NOME_BASE_DEST) REFERENCES
BASE_ESPACIAL(PLANETA, NOME) ON DELETE CASCADE
);

```

A tabela MISSAO_EXPLORACAO foi feita para armazenar informações sobre missões de exploração, mantendo a integridade referencial com as tabelas MISSAO_ESPACIAL, NAVE e BASE_ESPACIAL, e possibilitando a exclusão automática de registros quando as entidades associadas eram removidas. A FK_MISSAO_EXPLORACAO1 é uma restrição de chave estrangeira que faz referência à coluna MISSAO da tabela MISSAO_ESPACIAL, se uma missão espacial vinculada à missão de exploração fosse removida, a missão de exploração também seria eliminada. A FK_MISSAO_EXPLORACAO2 é uma restrição de chave estrangeira que faz referência à

coluna NAVE da tabela NAVE, e garante que se a nave vinculada à missão de exploração fosse removida, a missão de exploração também seria excluída. A FK_MISSAO_EXPLORACAO3 é uma restrição de chave estrangeira que faz referência às colunas PLANETA_BASE_DEST e NOME_BASE_DEST da tabela BASE_ESPACIAL, e certifica que se a base espacial vinculada à missão de exploração fosse removida, a missão de exploração também seria excluída.

```
CREATE TABLE MISSAO_COMERCIO (
    MISSAO NUMBER(15),
    VERBA_MAXIMA NUMBER(15),
    NAVE NUMBER(15) NOT NULL,
    PLANETA_BASE_DEST NUMBER(15) NOT NULL,
    NOME_BASE_DEST VARCHAR2(50) NOT NULL,
    CONSTRAINT PK_MISSAO_COMERCIO PRIMARY KEY (MISSAO),
    CONSTRAINT FK_MISSAO_COMERCIO1 FOREIGN KEY (MISSAO)
REFERENCES MISSAO_ESPACIAL(ID) ON DELETE CASCADE,
    CONSTRAINT FK_MISSAO_COMERCIO2 FOREIGN KEY (NAVE)
REFERENCES NAVE(ID) ON DELETE CASCADE,
    CONSTRAINT FK_MISSAO_COMERCIO3 FOREIGN KEY
(PLANETA_BASE_DEST, NOME_BASE_DEST) REFERENCES
BASE_ESPACIAL(PLANETA, NOME) ON DELETE CASCADE
);
```

A tabela MISSAO_COMERCIO foi projetada para armazenar informações sobre missões de comércio, mantendo integridade referencial com as tabelas MISSAO_ESPACIAL, NAVE e BASE_ESPACIAL, e permitindo a exclusão automática de registros quando as entidades relacionadas eram excluídas. A FK_MISSAO_COMERCIO1 é uma restrição de chave estrangeira que referenciava a coluna MISSAO da tabela MISSAO_ESPACIAL, essa restrição foi configurada para exclusão em cascata, o que significava que se uma missão espacial associada à missão de comércio fosse excluída, a missão de comércio também seria excluída. A FK_MISSAO_COMERCIO2 é uma restrição de chave estrangeira que referenciava a coluna NAVE da tabela NAVE, essa restrição estava configurada para exclusão em cascata, assegurando que se a nave associada à missão de comércio fosse excluída, a missão de comércio também seria excluída. A FK_MISSAO_COMERCIO3 é uma restrição de chave estrangeira que referenciava as colunas PLANETA_BASE_DEST e NOME_BASE_DEST da tabela BASE_ESPACIAL. Essa restrição estava configurada para exclusão em cascata (ON DELETE CASCADE), garantindo que se a base espacial associada à missão de comércio fosse excluída, a missão de comércio também seria excluída.

```
CREATE TABLE MISSAO_COLONIZACAO (
    MISSAO NUMBER(15),
    NAVE NUMBER(15) NOT NULL,
    PLANETA NUMBER(15) NOT NULL,
    CONSTRAINT PK_MISSAO_COLONIZACAO PRIMARY KEY (MISSAO),
    CONSTRAINT FK_MISSAO_COLONIZACAO1 FOREIGN KEY (MISSAO)
REFERENCES MISSAO_ESPACIAL(ID) ON DELETE CASCADE,
```

```

        CONSTRAINT FK_MISSAO_COLONIZACAO2 FOREIGN KEY (NAVE)
REFERENCES NAVE(ID) ON DELETE CASCADE,
        CONSTRAINT FK_MISSAO_COLONIZACAO3 FOREIGN KEY (PLANETA)
REFERENCES PLANETA(ID) ON DELETE CASCADE
);

```

A tabela foi projetada para armazenar informações sobre missões de comércio e colonização, garantindo integridade referencial com as tabelas MISSAO_ESPACIAL, NAVE e PLANETA, e permitindo a exclusão automática de registros quando as entidades relacionadas são excluídas. FK_MISSAO_COLONIZACAO1 é uma restrição de chave estrangeira que referencia a coluna MISSAO da tabela MISSAO_ESPACIAL, essa restrição está configurada para exclusão em cascata, o que significa que se uma missão espacial associada à missão de colonização for excluída, a missão de colonização também será excluída. FK_MISSAO_COLONIZACAO2 é uma restrição de chave estrangeira que referencia a coluna NAVE da tabela NAVE, e com CASCADE assegurando que se a nave associada à missão de colonização for excluída, a missão de colonização também será excluída. FK_MISSAO_COLONIZACAO3 é uma restrição de chave estrangeira que referencia a coluna PLANETA da tabela PLANETA garantindo que se o planeta associado à missão de colonização for excluído, a missão de colonização também será excluída.

```

CREATE TABLE COMERCIO_PRODUTO (
    MISSAO NUMBER(15),
    NOME_PRODUTO VARCHAR(50),
    ORIGEM_PRODUTO VARCHAR2(50),
    NOME_CIV VARCHAR2(50),
    CATEG_CIV CHAR(1),
    TIPO_COMERCIO VARCHAR2(50) NOT NULL,
    QUANTIDADE NUMBER(15) NOT NULL,
    CONSTRAINT PK_COMERCIO_PRODUTO PRIMARY KEY (MISSAO,
NOME_PRODUTO, ORIGEM_PRODUTO, NOME_CIV, CATEG_CIV),
    CONSTRAINT FK_COMERCIO_PRODUTO1 FOREIGN KEY (MISSAO)
REFERENCES MISSAO_COMERCIO(MISSAO) ON DELETE CASCADE,
    CONSTRAINT FK_COMERCIO_PRODUTO2 FOREIGN KEY (NOME_CIV,
CATEG_CIV, NOME_PRODUTO, ORIGEM_PRODUTO) REFERENCES
PRODUTO_CIVILIZACAO(NOME_CIV, CATEG_CIV, NOME_PRODUTO,
ORIGEM_PRODUTO) ON DELETE CASCADE
);

```

A tabela COMERCIO_PRODUTO foi projetada para armazenar informações sobre o comércio de produtos entre missões, civilizações e produtos, mantendo integridade referencial com as tabelas MISSAO_COMERCIO e PRODUTO_CIVILIZACAO e permitindo a exclusão automática de registros quando as entidades relacionadas são excluídas. PK_COMERCIO_PRODUTO é uma restrição de chave primária composta pelas colunas MISSAO, NOME_PRODUTO, ORIGEM_PRODUTO, NOME_CIV e CATEG_CIV, garantindo a unicidade dessa combinação. FK_COMERCIO_PRODUTO1 é uma restrição de chave estrangeira que referencia a coluna MISSAO da tabela

MISSAO_COMERCIO , por ser projetada com ON DELETE CASCADE se uma missão de comércio associada a um produto for excluída, os registros correspondentes na tabela COMERCIO_PRODUTO também serão excluídos.

```
CREATE TABLE EQUIPAMENTOS(
    MISSAO_EXPLORACAO NUMBER(15),
    EQUIPAMENTO VARCHAR2(50),
    CONSTRAINT PK_EQUIPAMENTOS PRIMARY KEY
(MISSAO_EXPLORACAO, EQUIPAMENTO),
    CONSTRAINT FK_EQUIPAMENTOS FOREIGN KEY
(MISSAO_EXPLORACAO) REFERENCES MISSAO_EXPLORACAO(MISSAO) ON
DELETE CASCADE
);
```

A tabela EQUIPAMENTOS foi desenvolvida para armazenar dados relacionados aos equipamentos associados a missões de exploração. Essa estrutura garante a integridade referencial com a tabela MISSAO_EXPLORACAO , assegurando uma conexão consistente. Adicionalmente, está configurada para possibilitar a exclusão automática de registros de equipamentos quando a missão de exploração correspondente é excluída, mantendo a coerência nos dados.

12. Alimentação Inicial da Base de Dados:

No contexto de um cenário espacial fictício, diversos dados foram inseridos para representar informações relacionadas a sistemas planetários, estrelas, planetas, satélites naturais, civilizações, rotas interestelares, missões espaciais (comércio, colonização e exploração), produtos comerciais, equipamentos e ameaças potenciais. As missões espaciais envolvem naves, bases espaciais e planetas, enquanto o comércio entre civilizações está associado a produtos específicos. Missões de colonização e exploração são relacionadas, incluindo detalhes sobre equipamentos utilizados. Esse conjunto de dados permite explorar as dinâmicas complexas e interconectadas de um universo fictício, proporcionando uma variedade de elementos, todos respeitando as restrições impostas na criação das tabelas.

13. Consultas:

Em seguidas estão descritas as consultas solicitadas; uma descrição melhor que cada parte da consulta faz pode ser encontrada no arquivo queries.sql fornecido, no qual os comentários indicam melhor o funcionamento das consultas.

- **Contar quantas ameaças tem nas rotas de cada missão:**

Essa consulta conta quantas ameaças de rota ou de planeta cada missão encontra ao longo do seu trajeto.

```
SELECT NOME, COUNT(*) AS NUM_AMEACAS FROM (SELECT NOME,
NOME_AMEACA AS AMEACA
FROM MISSAO_ESPACIAL M JOIN ROTA_MISSAO RM ON M.ID = RM.MISSAO
JOIN AMEACA_ROTA AR ON RM.ORIGEM_ROTA = AR.ORIGEM_ROTA AND
RM.DESTINO_ROTA = AR.DESTINO_ROTA
UNION
SELECT NOME, AMEACA
FROM MISSAO_ESPACIAL M JOIN ROTA_MISSAO RM ON M.ID = RM.MISSAO
JOIN AMEACA_PLANETA AP ON AP.PLANETA = RM.ORIGEM_ROTA OR
AP.PLANETA = RM.DESTINO_ROTA)
GROUP BY NOME
```

- **Listar civilizações que comercializam todos os produtos que uma missão comercializa (independente de ser compra ou venda):**

Essa consulta é de divisão relacional. Escolhemos como referência a missão 1. O resultado da consulta é a lista das civilizações que comercializam todos os produtos que a missão de ID 1 comercializa.

```
SELECT NOME FROM CIVILIZACAO C1 WHERE NOT EXISTS (
(SELECT NOME_PRODUTO FROM MISSAO_ESPACIAL M JOIN
COMERCIO_PRODUTO CP ON M.ID = CP.MISSAO WHERE M.ID = 1)
MINUS
(SELECT NOME_PRODUTO FROM CIVILIZACAO C JOIN
PRODUTO_CIVILIZACAO PC ON C.NOME = PC.NOME_CIV AND C.CATEGORIA
= PC.CATEG_CIV WHERE C.NOME = C1.NOME AND C.CATEGORIA =
C1.CATEGORIA)
)
```

- **Contar quantas missões cada nave realizou:**

Essa consulta conta quantas missões cada nave realizou.

```
SELECT NV.NOME, COUNT(NAVE) AS NUM_MISSOES FROM (SELECT
MCOM.MISSAO, NCOM.NAVE FROM MISSAO_COMERCIO MCOM JOIN
NAVE_CARGA NCOM ON NCOM.NAVE = MCOM.NAVE
UNION
SELECT MEXP.MISSAO, NCIEN.NAVE FROM MISSAO_EXPLORACAO MEXP
JOIN NAVE_CIENTIFICA NCIEN ON NCIEN.NAVE = MEXP.NAVE
UNION
SELECT MCOL.MISSAO, NCOL.NAVE FROM MISSAO_COLONIZACAO MCOL
JOIN NAVE_COLONIZADORA NCOL ON NCOL.NAVE = MCOL.NAVE)
RIGHT JOIN NAVE NV ON NAVE = NV.ID
```


GROUP BY NV.NOME

- Informar o valor total vendido por cada missão de comércio:

Essa consulta retorna o valor total vendido por cada missão de comércio.

```
SELECT ME.NOME, SUM(P.PRECO * CP.QUANTIDADE) AS VALOR_VENDAS
FROM MISSAO_ESPACIAL ME JOIN MISSAO_COMERCIO MC ON ME.ID =
MC.MISSAO
JOIN COMERCIO_PRODUTO CP ON CP.MISSAO = MC.MISSAO
JOIN PRODUTO P ON P.NOME = CP.NOME_PRODUTO AND P.ORIGEM =
CP.ORIGEM_PRODUTO
WHERE CP.TIPO_COMERCIO = 'Venda'
GROUP BY ME.NOME
```

- Listar as missões de todos os tipos com suas naves correspondentes caso a missão já tenha uma nave designada:

Essa consulta retorna uma lista de todas as missões do sistema com as suas naves correspondentes caso a missão já tenha uma nave designada.

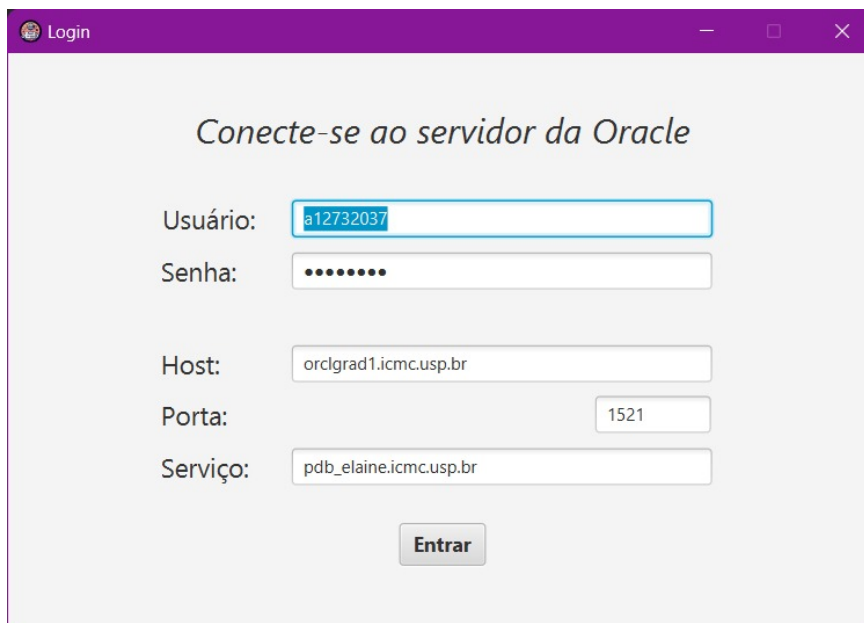
```
SELECT ME.NOME AS MISSAO, NV.NOME AS NAVE FROM
MISSAO_ESPACIAL ME LEFT JOIN
(SELECT MCOM.MISSAO, NCOM.NAVE AS NAVE_MISSAO FROM
MISSAO_COMERCIO MCOM JOIN NAVE_CARGA NCOM ON NCOM.NAVE =
MCOM.NAVE
UNION
SELECT MEXP.MISSAO, NCEN.NAVE AS NAVE_MISSAO FROM
MISSAO_EXPLORACAO MEXP JOIN NAVE_CIENTIFICA NCEN ON
NCEN.NAVE = MEXP.NAVE
UNION
SELECT MCOL.MISSAO, NCOL.NAVE AS NAVE_MISSAO FROM
MISSAO_COLONIZACAO MCOL JOIN NAVE_COLONIZADORA NCOL ON
NCOL.NAVE = MCOL.NAVE)
ON ME.ID = MISSAO
LEFT JOIN NAVE NV ON NV.ID = NAVE_MISSAO
ORDER BY ME.NOME, NV.NOME
```

14. Implementação do Sistema:

O protótipo, desenvolvido em Java e com interface gráfica utilizando a biblioteca JavaFX, é uma ferramenta para o gerenciamento de informações sobre planetas e missões espaciais. Ele se conecta a uma base de dados Oracle, permitindo aos usuários realizar operações de cadastro e consulta de Planetas, além de consultas de Missões Espaciais.

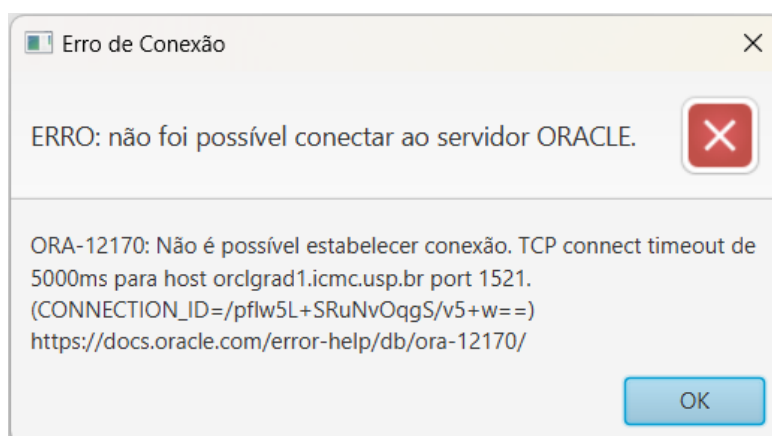
14.1. Login:

Ao iniciar o aplicativo, os usuários são recebidos por uma tela de login, que para fins práticos está fixo no login da nossa base de dados.

A imagem mostra a interface de login de uma aplicação. No topo, há uma barra de título com o ícone de uma pessoa e o texto "Login". O conteúdo principal da janela tem o título "Conecte-se ao servidor da Oracle". Abaixo dele, há campos de entrada para "Usuário:" (contendo "a12732037"), "Senha:" (com pontos para ocultar), "Host:" (contendo "orclgrad1.icmc.usp.br"), "Porta:" (contendo "1521") e "Serviço:" (contendo "pdb_elaine.icmc.usp.br"). Um botão "Entrar" está posicionado abaixo dos campos.

Tela de login da aplicação.

Se a conexão falhar, o usuário receberá um aviso de erro indicando que a conexão com o servidor do banco de dados não foi bem-sucedida.



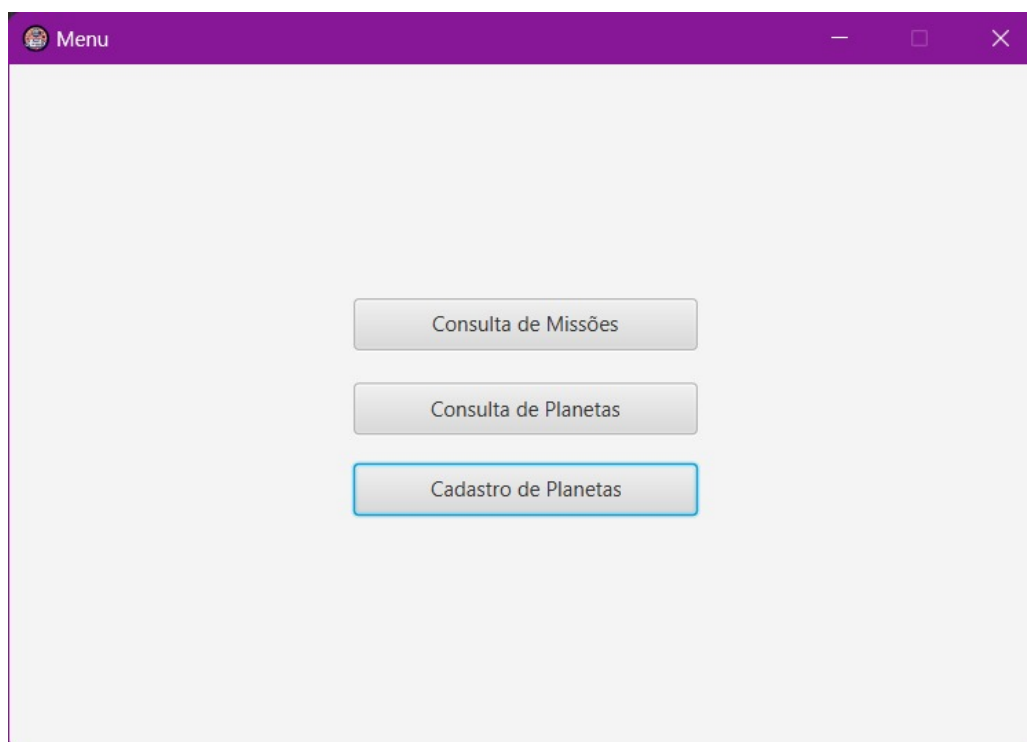
Tela de erro de conexão.

14.2. Opções:

Uma vez que a conexão é estabelecida, três opções principais são apresentadas:

1. **Consulta de Missões:** Esta funcionalidade permite aos usuários pesquisar missões com base na base espacial, no nome da missão e na data em que foi realizada.
2. **Consulta de Planetas:** Os usuários podem pesquisar um planeta específico com base no sistema planetário em que está inserido e no seu nome.

3. **Cadastro de Planetas:** Esta opção permite aos usuários inserir novos planetas no banco de dados, preenchendo campos com todos os atributos relevantes de um planeta, incluindo o sistema planetário e a galáxia.



Tela de opções.

14.3. Consulta de Missões:

Ao escolher a opção '**Consulta de Missões**', o usuário é direcionado para uma nova tela. Nesta tela, é possível pesquisar missões espaciais previamente cadastradas na base de dados, utilizando critérios como a Base de Origem, o Nome da Missão e/ou a Data de Início da Missão. Se o usuário não fornecer nenhum desses dados, todas as missões cadastradas serão exibidas.

Consultar Missões

Base origem: Nome Missão: Data início:

Buscar

Planeta	Fim	Descrição	Tripulação	Objetivo	Duração	Nível Perigo
1	969-07-2...	Pousar na ...	3	ExploraÃs...	8	7
2	2023-11-3...	ExploraÃs...	0	ExploraÃs...	0	5
3	2150-03-1...	ExploraÃs...	50	ExploraÃs...	5	9
4	2200-06-0...	ExploraÃs...	8	ExploraÃs...	3	8
5	2202-11-1...	Comerciali...	5	Comerciali...	2	3

Voltar

Tela de consulta de missões.

Caso o usuário insira uma data em formato inválido ou um texto que não corresponda a uma data no campo 'Data de Início', um aviso de erro será exibido, informando que um valor inválido foi inserido no campo de Data.

Erro na data.

Você inseriu um valor inválido na data.

Text 'texto invalido' could not be parsed at index 0

OK

Tela erro na data.

14.4. Consultar Planetas:

Quando a opção '**Consulta de Planetas**' é selecionada, o usuário é encaminhado para um novo ambiente. Neste espaço, ele pode explorar os planetas já registrados no banco de dados, usando como critérios o Sistema Planetário e/ou o Nome do Planeta. Caso nenhuma dessas informações seja fornecida pelo usuário, a lista completa de planetas registrados será apresentada.

Consultar planetas

Consultar Planetas

Sistema Planetários:

Nome Planeta:

Buscar

ID	Sistema	Nome	Temperatura	Pressão	Clima
1	Sistema Solar	Terra	20.0	1013.0	Tempe...
2	Sistema Solar	Marte	-55.0	636.0	Frio
3	Alpha Centauri	MineWorld	1200.0	500.0	Quente
4	Trappist-1	CraftWorld	-40.0	700.0	Frio
5	Sistema Gliese-163	Gliese-163 B	-78.0	452.0	Frio
6	Alpha Centauri	você	0.0	0.0	
7	Kepler-186	josé	0.0	0.0	

Voltar

Tela de consulta de planetas.

14.5. Cadastrar Planeta:

Ao optar por '**Cadastro de Planetas**', o usuário é levado a um novo ambiente. Neste local, ele tem a oportunidade de adicionar novos planetas ao banco de dados, associando-os a um Sistema Planetário que já esteja registrado.

Cadastrar Planetas

Cadastrar Planeta

Nome do Planeta:

Sistema Planetário:

Temperatura:

Pressão:

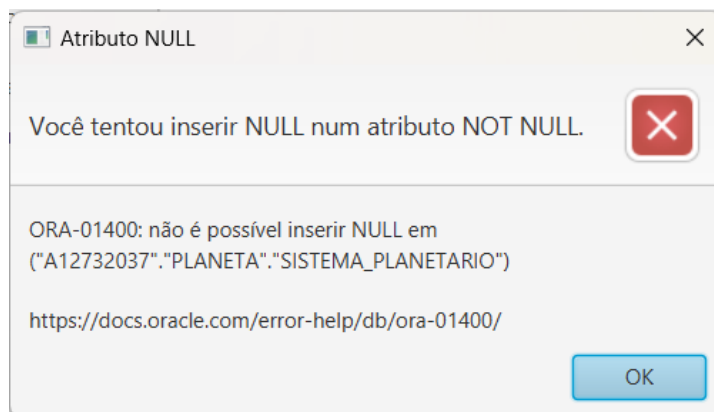
Clima:

Cadastrar

Voltar

Tela de cadastro de planetas.

Se durante o cadastro, os campos 'Planeta' e/ou 'Sistema Planetário', que são marcados como '**NOT NULL**' no banco de dados, estiverem em branco, o usuário receberá um aviso indicando que esses campos não podem ser deixados vazios.



Tela de erro atributo **NULL**

14.6. Integração Oracle SGDB - JAVA:

O aplicativo utiliza a biblioteca JDBC para estabelecer uma conexão com a conta Oracle e o Java SQL para realizar operações de atualização ou consulta no banco de dados.

14.6.1. Insert, Delete, Update:

As operações de atualização, que incluem a inserção, exclusão e modificação de tabelas, são executadas utilizando a função **Connection.prepareStatement(String sql)**. Esta função prepara a instrução SQL para execução, garantindo que ela seja processada de maneira eficiente e segura.

A execução da instrução de atualização é realizada através da função **PreparedStatement.executeUpdate()**. Esta função é responsável por aplicar as alterações no banco de dados, seja inserindo novos registros, excluindo registros existentes ou modificando os dados de registros existentes.

É crucial observar que após a execução de operações de inserção, remoção ou modificação, é necessário confirmar a execução da instrução com **Connection.commit()**. Este passo é essencial para garantir que as alterações sejam permanentemente aplicadas ao banco de dados.

No entanto, se ocorrer uma exceção durante a inserção, como uma violação de chave ou uma inserção não intencional de **NULL**, é necessário reverter a operação com **Connection.rollback()**. Esta função desfaz todas as alterações feitas na transação atual, preservando a integridade dos dados no banco de dados.

```

1 String nome = tfPlaneta.getText();
2 String clima = tfClima.getText();
3
4 PreparedStatement obterID = conexao.prepareStatement("SELECT MAX (ID) + 1 FROM PLANETA");
5 ResultSet r = obterID.executeQuery();
6 int id = 1;
7 while (r.next())
8     id = r.getInt(1);
9 r.close();
10
11 PreparedStatement linha = conexao.prepareStatement("INSERT INTO PLANETA values (?, ?, ?, ?, ?, ?, ?)");
12 linha.setInt(1, id);
13 linha.setString(2, sistema);
14 linha.setString(3, galaxia);
15 linha.setString(4, nome);
16
17 if (tfTemperatura.getText().isEmpty())
18     linha.setNull(5, Types.FLOAT);
19 else
20     linha.setFloat(5, Float.parseFloat(tfTemperatura.getText()));
21
22 if (tfPressao.getText().isEmpty())
23     linha.setNull(6, Types.FLOAT);
24 else
25     linha.setFloat(6, Float.parseFloat(tfPressao.getText()));
26
27 linha.setString(7, clima);
28 linha.executeUpdate();
29
30 conexao.commit();

```

Código exemplo de Inserção - *CadastrarPlanetasController.java* - método “*cadastrar*”.

14.6.2. Consultas:

As operações de consulta, que são essenciais para a interação com o banco de dados, mas que não alteram a estrutura da tabela, são realizadas utilizando a função ***PreparedStatement.executeQuery()***. Esta função é uma parte crucial da biblioteca Java SQL, pois permite a execução de consultas SQL de maneira eficiente e segura.

Ao ser executada, a função ***PreparedStatement.executeQuery()*** retorna um objeto ***ResultSet***. Este objeto contém os resultados da consulta SQL, organizados de uma maneira que permite ao usuário do aplicativo navegar pelos dados. Cada linha do ***ResultSet*** corresponde a um registro do banco de dados que satisfaz os critérios da consulta.

```

1      String linhaSQL = "SELECT * FROM MISSAO_ESPACIAL ";
2
3      ArrayList atributosLinha = new ArrayList();
4
5      if(baseOrigem != null){
6          if(!linhaSQL.contains("WHERE"))
7              linhaSQL += "WHERE ";
8          else
9              linhaSQL += "AND ";
10         linhaSQL += "PLANETA_BASE = ? AND NOME_BASE LIKE ? ";
11         atributosLinha.add(baseOrigem.getPlaneta());
12         atributosLinha.add(baseOrigem.getNome());
13     }
14
15     if(!nomeMissao.isEmpty()){
16         if(!linhaSQL.contains("WHERE"))
17             linhaSQL += "WHERE ";
18         else
19             linhaSQL += "AND ";
20         linhaSQL += "NOME LIKE ? ";
21         atributosLinha.add(nomeMissao);
22     }
23
24     if(dataMissao != null){
25         if(!linhaSQL.contains("WHERE"))
26             linhaSQL += "WHERE ";
27         else
28             linhaSQL += "AND ";
29         linhaSQL += "DATA_INICIO LIKE TO_DATE(?,\''dd-mm-yyyy'\')";
30         String data = dataMissao.getDayOfMonth() + "-" + dataMissao.getMonthValue() + "-" + dataMissao.getYear();
31         System.out.println(data);
32         atributosLinha.add(data);
33     }
34
35     System.out.println(linhaSQL);
36
37     PreparedStatement linha = conexao.prepareStatement(linhaSQL);
38     for(int i = 0; i < atributosLinha.size(); i++){
39         if (atributosLinha.get(i) instanceof String)
40             linha.setString(i+1, (String) atributosLinha.get(i));
41         else
42             linha.setInt(i+1, (int) atributosLinha.get(i));
43     }
44
45     ResultSet resultSet = linha.executeQuery();

```

Código exemplo de consulta - *ConsultarMissoesController.java* - método “*buscar*”.

14.7. Tratamento de exceções:

As exceções consideradas relevantes para o usuário final são apresentadas em janelas de alerta, permitindo que o usuário seja notificado sobre qualquer erro que tenha ocorrido ou sido cometido. Quaisquer outras exceções são gerenciadas e/ou exibidas no terminal para análise e resolução futura.



Código exemplo de tratamento de Exceção - *ConsultarMissoesController.java* - método "buscar".

15. Conclusão:

Por meio deste projeto foi possível compreender o processo de desenvolvimento de uma base de dados desde o estudo de requisitos até a implementação. Ficou evidente o quanto a fase de projeto, sobretudo o desenvolvimento dos diagramas MER e Modelo Relacional, facilitam a fase de implementação. O desenvolvimento desses modelos sistematizados previne a ocorrência de diversos tipos de inconsistências dentro dos limites de cada um deles. No fim, o processo de criar a base de dados e alimentá-la foi mecânico dado todo o trabalho feito nas etapas anteriores.